НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»

УДК _____

«До захисту допущено»

Завідувач кафедри

_____ *Стіренко С.Г.*
(підпис) (ініціали, прізвище)

“____” _____ 2020 р.

Магістерська дисертація

зі спеціальності: *121. Інженерія програмного забезпечення*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *121. Інженерія програмного забезпечення комп’ютерних систем*

на тему: Оптимізована рекомендаційна система з використанням технології нейронних мереж

Виконала: студентка 6 курсу, групи ІП-з92мп
(шифр групи)

Литвиненко Анна Сергіївна
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц.,к.т.н., с.н.с.Долголенко О.М
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доцент кафедри АУТС, к.т.н., Андрій Писаренко
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
« » 2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Литвиненко Анні Сергіївні
(прізвище, ім'я, по батькові)

1. Тема дисертації Оптимізована рекомендаційна система з використанням технології нейронних мереж
Науковий керівник дисертації Долголенко О.М доц.,к.т.н., с.н.с.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26» жовтня *2020* р. № 3133

2. Строк подання студентом дисертації 25 листопада 2020 р.

3. Об'єкт дослідження Підходи до подолання проблеми «холодного старту» у рекомендаційних системах

4. Предмет дослідження Розробка рекомендаційної системи, що буде вирішувати проблему «холодного старту» шляхом аналізу графічних зображень та з використанням технології нейронних мереж

5. Перелік завдань, які потрібно розробити: проаналізувати існуючі рішення подолання проблеми «холодного старту», проаналізувати архітектуру рекомендаційних систем, розглянути принципи технології нейронних мереж та вибрати найбільш подходящу архітектуру, розробити алгоритм, розробити прототип програмного забезпечення та тестове середовище для нього, представити стартап-проект до розроблюваного програмного продукту

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1. Дослідження принципів роботи рекомендаційних систем та нейронних мереж	Долголенко О.М доц.,к.т.н., с.н.с.		
Розділ 2. Вибір засобів вирішення задачі та розробка алгоритму			
Розділ 3. Реалізація вирішення задачі та розробка програми			
Розділ 4. Розробка стартап-проекту			

7. Дата видачі завдання 18.12.2019

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Затвердження теми роботи	18.12.2019	
2.	Вивчення та аналіз завдання	20.05.2020 – 02.06.2020	
3.	Пошук матеріалів по темі роботи	03.06.2020 – 20.06.2020	
4.	Розробка програмного продукту	3.08.2020 – 31.08.2020	
5.	Оформлення пояснювальної записки	1.09.2020 – 20.11.2020	
6.	Передзахист	25.11.2020	
7.	Захист	17.12.2020	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Оптимізована рекомендаційна система з використанням технології нейронних мереж

студенткою: Литвиненко Анною Сергіївною

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 83 аркуші основного тексту, 34 ілюстрації, 23 таблиць. При підготовці використовувалася література з 20 різних джерел.

Актуальність. З кожним днем у різні сервіси з продажу впроваджуються аналітичні додатки, що допомагають краще зрозуміти користувача та запропонувати придбати товар, що буде відповідати його вподобанням. Таким підбором рекомендацій займаються рекомендаційні системи. На початку їх запуску дуже часто не вистачає початкових даних про користувачів, щоб побудувати рекомендації правильно, тому існує проблема «холодного старту».

Використання технологій нейромереж не є новим, але має велику популярність через доволі точні розрахунки та гнучкість у налаштуванні. Розробка рекомендаційної системи, що може подолати проблему «холодного старту» та використовує технологію нейронних мереж є актуальною.

Мета і завдання дослідження. Метою даної магістерської роботи є розробка рекомендаційної системи, що вирішує проблему «холодного старту» шляхом аналізу графічних зображень з використанням технології нейронних мереж.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- дослідження структури та принципів роботи рекомендаційних систем;
- дослідження видів архітектур та принципів роботи нейронних мереж, а особливо згорткових;
- побудова алгоритму роботи рекомендаційної системи;

- розробка прототипу рекомендаційної системи та тестового середовища для демонстрації результатів;
- ілюстрація роботи моделі та аналіз отриманих результатів;

Об’єкт дослідження – підходи до подолання проблеми «холодного старту» у рекомендаційних системах.

Предмет дослідження – Розробка рекомендаційної системи, що буде вирішувати проблему «холодного старту» шляхом аналізу графічних зображень та з використанням технології нейронних мереж.

Методи досліджень. Для досягнення поставлених в магістерській роботі задач, використано технології згорткових нейронних мереж.

Наукова новизна одержаних результатів роботи полягає у наступному:

- запропоновано вирішення проблеми «холодного старту» шляхом аналізу контенту товару з сервісу продажу товарів, що базується на аналізі графічного зображення товару з допомогою використання технології згорткових нейронних мереж.
- розроблено програмний продукт для побудови рекомендацій та тестове середовище для демонстрації результату.

Проведене дослідження дає можливість використання розробленої рекомендаційної системи для побудови рекомендацій товарів без потреби у персональних даних користувачів та одразу після запуску сервісу.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до вирішення задачі розробки рекомендаційної системи, що вирішує проблему «холодного старту» з використанням технології нейронних мереж. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Практична цінність. Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- розробка рекомендаційних систем;

- вдосконалення технології побудови рекомендацій на основі графічних зображень;
- аналіз схожості графічних зображень з використанням технології нейронних мереж;

Ключові слова

Рекомендаційні системи, проблема «холодного старту», нейронні мережі, згорткові нейронні мережі, аналіз графічних зображень, побудова рекомендацій.

ABSTRACT

on master's thesis

made on the topic: Neural network-based optimized

recommendation system

by: Anna Lytvynenko

The master thesis consists of introduction and four sections. Total volume of work: 83 sheets of the main text, 34 illustrations, 23 tables. Literature from 20 different sources was used in the preparation.

Relevance of the topic. Every day in various sales services are implemented analytical applications that help to better understand the user and offer to buy a product that will match his preferences. Recommendation systems are engaged in such selection of recommendations. At the beginning of their launch, there is often a lack of initial user data to build recommendations correctly, so there is a "cold start" problem.

The usage of neural network technology is not new, but it is very popular due to fairly accurate calculations and flexibility in configuration. The development of a recommendation system that can overcome the problem of "cold start" and uses neural network technology is relevant in nowadays.

The purpose of research. The purpose of this master's thesis is to develop a recommendation system that solves the problem of "cold start" by analyzing graphic images using neural network technology.

To achieve the goal of the study, the following tasks were set and solved:

• study the structure and operation principles of recommendation systems;

• study the types of architectures and principles of neural networks, especially convolutional;

• construction of the algorithm of the recommendation system;

• development of a prototype of a recommendation system and test environment to demonstrate the results;

• illustration of the model and analysis of the obtained results;

The object of research - approach to overcoming the problem of "cold start" in recommendation systems.

Subject of research - development of a recommendation system that will solve the problem of "cold start" by analyzing graphic images and using neural network technology.

Research methods. To achieve the objectives set in the master's thesis, the technology of convolutional neural networks is used.

The scientific novelty of the obtained results is as follows:

• the solution of the problem of "cold start" by the analysis of the content of the goods from the sale services based on the analysis of the graphic image of the goods with the technology of the convolutional neural networks is offered.

• developed a software product to create recommendations and a test environment to demonstrate the result.

The study makes it possible to use the developed recommendation system to build product recommendations without the need for personal data of users and immediately after the first launch of the service.

Personal contribution of the applicant. The master's research is a self-performed work, which reflects the personal author's approach and personally obtained theoretical and applied results related to the problem of developing a recommendation system that solves the problem of "cold start" using neural network technology. The formulation of the purpose and objectives of the study was carried out jointly with the supervisor.

Practical value. The obtained results can be used in future research in the following areas:

• development of recommendation systems;

• improving the technology of constructing recommendations based on graphic images;

• analysis of similarity of graphic images using neural network technology;

Keywords

Recommendation systems, the problem of "cold start", neural networks, convolutional neural networks, analysis of graphic images, construction of recommendations.

Пояснювальна записка
до магістерської дисертації

на тему: «Оптимізована рекомендаційна система з використанням технології нейронних мереж»

Київ – 2020

ЗМІСТ

ВСТУП

Інформаційні технології відіграють дуже важливу роль у сучасному світі. Кожного за нас повсякчас супроводжують новітні технології, а іноді навіть ті, про які ми не замислюємося. Інформатизація охоплює усі сфери нашого життя, починаючи з спілкування, рутинних щоденних справ, розважальної діяльності та, завершуючи, наукову діяльність.

Якщо розглядати будь-яку сферу, то можна дійти висновку, що чим більше вона розвивається, тим більше стає потреб. Тепер, навіть, вибір будь-чого можна доручити комп'ютерній системі. Такі системи є рекомендуючими і у наш час дуже поширені. Рекомендаційні системи є невід'ємною частиною великої кількості інтернет-магазинів, сервісів з онлайн перегляду фільмів, пошуку маршрутів для подорожей, прослуховування музики та новин, що будуть цікавими певному користувачу. Системи з рекомендацій товарів та послуг забезпечують бізнесу більший прибуток, адже може «вгадувати» бажання клієнта, чим привертає більше уваги та довіри.

Рекомендаційні системи працюють на основі аналізу даних користувачів, їх уподобань, маршрутів переходу на сайті та інше. Кожен користувач хоче, щоб йому рекомендували саме те, що він хоче отримати, придбати і для цього потрібно певним чином проаналізувати усі дані, що з ним зв'язані. Існуючі системи здебільшого розраховані на велику кількість вхідних статистичних даних, що вже була зібрана з допомогою користувачів протягом певного часу роботи сервісу. Але кожен сервіс, починаючи свою роботу, не має на старті вхідних аналітичних даних, щоб система рекомендації працювала коректно. Цей процес називається «холодним стартом».

Інтернет-магазини, що продають одяг та взуття, частіше за все можуть рекомендувати користувачу нові товари, базуючись на кольоровій гаммі, типу товару та його зображенні. Отже, для створення навіть самої простої системи рекомендацій потрібно проаналізувати зображення, щоб підібрати максимально подібні до того, яким цікавиться користувач.

Серед сучасних технологій обробки та аналізу даних можна виділити технологію нейронних мереж. Вона виділяється своєю відносною точністю та масовістю способів застосування. Нейронні мережі мають різну архітектуру, яка задовольняє певні потреби. Існують декілька архітектур, що показують досить високі результати у обробці даних зображення та його аналізу. З допомогою технології нейромереж та певної її архітектури можна розробити оптимізовану рекомендаційну систему для «холодного» старту з можливістю аналізу зображень.

Тому, тема магістерської дисертації, а саме вирішення проблеми «холодного старту» з допомогою технології нейромереж у рекомендаційних системах є актуальною. Також досліджується концепція, принципи роботи рекомендаційних систем, їх види, способи аналізу вхідних даних та технологія нейромереж, складові нейромереж, аналіз архітектур та вибір найбільш підходящої для вирішення поставленої задачі.

1 РОЗДІЛ
ДОСЛІДЖЕННЯ ПРИНЦИПІВ РОБОТИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ТА НЕЙРОННИХ МЕРЕЖ

1.1 Рекомендаційна система

Рекомендаційна система - це підклас системи фільтрації інформації, яка передбачає "рейтинг" або "оцінку", яку користувач надасть товару. Також такі системи називають рекомендаційна платформа або «двигун». Такі системи в основному використовуються в комерційних програмах.

Рекомендаційні системи використовуються в різних сферах і найчастіше зустрічаються як генератори списків відтворення для відео та музичних послуг, таких як стримінгові сервіси, онлайн кінотеатри, рекомендатори продуктів для таких послуг, як інтернет магазини, або рекомендаторів вмісту для платформ соціальних медіа. Ці системи можуть працювати з одним типом контенту, наприклад музикою, або кількома типами на платформах та між ними, таких як новини, книги та пошукові запити. Існують також популярні системи рекомендацій для певних тем, таких як ресторани та знайомства в Інтернеті. Також були розроблені системи рекомендацій для вивчення дослідницьких статей та експертів, співавторів та фінансових служб.

Системи рекомендацій найчастіше використовують колаборативну фільтрацію, фільтрацію на основі змісту або їх поєднання.

Суть систем з колаборативною фільтрацією полягає у тому, що будується модуль на основі попередньої поведінки користувача (товари, що користувач придбав раніше, або оцінки, що користувач проставив цим товарам), а також схожої поведінки інших користувачів. Після збору даних модель використовується для прогнозування товарів, які можуть зацікавити користувача.

Системи, що базуються на вмісті використовують заздалегідь визначені характеристики товарів, щоб рекомендувати інші товари за схожими характеристиками.

1.2 Колаборативна фільтрація

Головною метою алгоритмів колаборативної фільтрації є рекомендація нових товарів, послуг або прогнозування цінності певного товару для конкретного користувача но основі його попередніх уподобань, або думок інших користувачів.

За типовим сценарієм колаборативного фільтрування існує список m користувачів $U = \{u_1, u_2, \dots, u_m\}$ та список елементів $I = \{i_1, i_2, \dots, i_n\}$. Кожен користувач u_i має список елементів $I_{u_i,}$ які він оцінив, або висловив свою думку про них. Оцінка товару може бути подана як рейтинговий бал, як правило, у межах певної числової шкали, або може бути отримана з допомогою аналізу переходів користувача за гіперпосиланнями на ці товари. Також, можливо, що I_{u_i} можу бути пустою I-множиною.

Задача алгоритмів колаборативної фільтрації допомогти знайти користувачу $u_a \in U$, названому як «активний користувач», товари, що йому можуть сподобатися. Інтерес до товарів можна показати з допомогою двох форм:

- Прогнозування – числове значення $P_{a,j}$, що відображає прогнозований інтерес елементу $i_j \notin I_{u_a}$ для активного користувача u_a. Це передбачення також підпадає під ту саму шкалу (наприклад, від 1 до 5), як з допомогою якої користувач u_a оцінював товари.
- Рекомендація – список, що складається з N елементів, $I_r \subset I$ що сподобаються користувачу найбільше. Рекомендації повинні базуватися тільки на елементах, що ще користувач не купував, тобто $I_r \cap I_{u_a} = \Phi$. Такий вишляд алгоритмів колаборативної фільтрації також відомий, як «TOP-N recommendation».

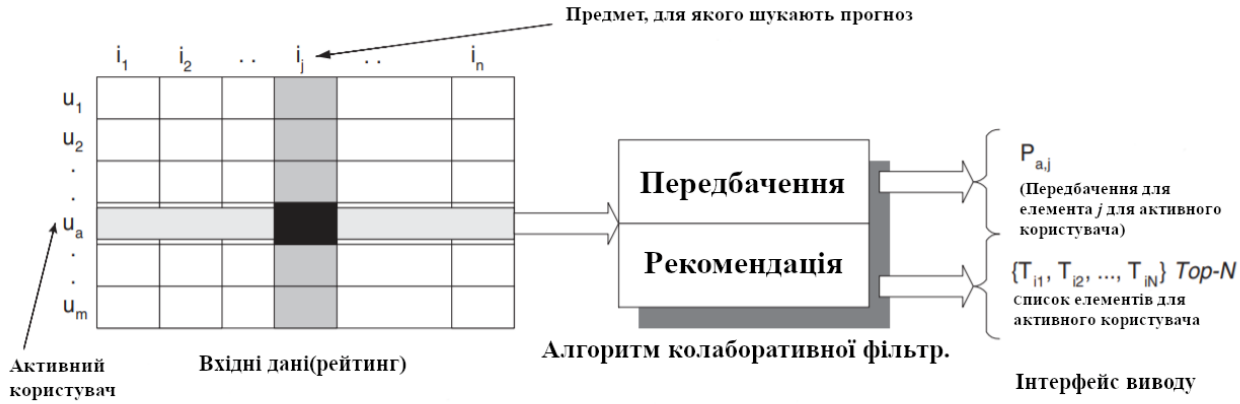

Рис. 1.1 Процес колаборативної фільтрації

На Рис. 1.1 можна побачити схематичну діаграму процесу колаборативної фільтрації. Алгоритм репрезентує матрицю $A\ m \times n$, що містить дані про користувача, елементи та їх оцінку. Кожне $a_{i,j}$ в A показує оцінку i-того користувача j-того елемента. Кожна окрема оцінка визначена в рамках певної числової шкали та може бути також дорівнювати 0, що буде означати, що користувач ще не оцінив цей елемент. Науковці поділили алгоритми колаборативної фільтрації на дві головні категорії – засновані на пам'яті(засновані на користувачі) та засновані на моделі(засновані на елементах) [1].

1.2.1 Алгоритми засновані на пам'яті

Алгоритми, які засновані на пам'яті використовують всю базу даних елементів користувача для створення прогнозу. Такі системи використовують статистичні методи для пошуку списку користувачів, відомих як «сусіди», що мають схожість з цільовим користувачем (тобто вони схоже оцінюють ті самі товари, або вони схильні придбати одні й ті самі товари). Коли сусідство двох користувачів встановлено, то системи використовують різні алгоритми для поєднання переваг сусідів для отримання прогнозу або рекомендації «TOP-N» для активного користувача. Такі техніки відомі як «найближчий сусід» або фільтрація, заснована на користувачах, найбільш популярні та широко використовуються на практиці.

Для того, щоб виміряти подібність, потрібно визначити співвідношення між двома користувачами. Таким чином визначається значення від -1 до 1, яку визначає, як схожі ці користувачі. Значення 1 означає, що користувачі оцінюють елемент

однаково, а значення -1 означає, що вони оцінюють товар по-різному (один оцінив позитивною, високою оцінкою, а інший - низькою) [2].

Коефіцієнт кореляції Пірсона може визначити подібність користувачів. Це основний алгоритм кореляції для зразків, адаптованих для оцінки інформації. Він намагається виміряти, наскільки обидва користувачі відрізняються від їхніх звичайних голосів (тобто величина голосів кожного у порівнянні з їх середньою оцінкою). Якщо виміри будуть однаково змінюватися, щодо загальних оцінок, то вони отримають позитивну кореляцію. Інакше, вони отримають негативну.

Інше вимірювання подібності називається векторною подібністю. Можна розглядати користувачів, як вектори у n-вимірному просторі, де n-кількість елементів у базі даних. Як у будь-яких двох векторів, ми можемо порівняти кут між ними. Інтуїтивно, якщо два вектори в цілому спрямовані в одному напрямку, вони отримують позитивну подібність. Якщо вони вказують в протилежні сторони, вони отримують негативну подібність. Щоб імітувати це, потрібно просто взяти косинус кута між цими двома векторами, що дає нам значення від -1 до 1.

Для передбачення рейтингу елемента для активного користувача, потрібно знайти всі ваги між активним користувачем та всіма іншими користувачами. Потім ми беруться всі ненульові ваги і пропонуємо користувачам "голосувати" за те, що, на їх думку, повинен оцінити актив. Ті, хто має більшу вагу, матимуть більше значення в процесі голосування. Після підрахунку цих голосів отримується передбачуване голосування.

Можна зауважити, що голосування базується на тому, наскільки далеко від середнього показника користувач оцінює елемент (товар) – тобто наскільки далеко від середнього показника активний користувач оцінить елемент. Таким чином, при позитивній кореляції активний користувач погоджується з тим, наскільки далеко інший користувач проголосував за певний пункт, а при негативній кореляції активний користувач не погоджується (тобто йде в протилежному напрямку) з голосом іншого користувача.

Такий алгоритм працює добре, але існують деякі вдосконалення такого алгоритму:

1) Голосування за замовчуванням. Кореляція, як вимірювання подібності, працює не дуже добре на розріджених наборах даних. Тобто, коли у двох користувачів мало спільних предметів, їх вага, як правило, надмірно підкреслюється. Голосування за замовчуванням просто додає ряд уявних пунктів, які оцінюються за однаковим шаблоном, щоб зробити голосування більш нейтральним.
2) Зворотна частота користувачів. Існує інтуїтивна оцінка, яка зазвичай віддає перевагу предметам, що мають менше значення ваги, ніж більш рідкісні предмети. Тобто, якщо всім сподобались «Зоряні війни», це не так допомагає у визначенні ваги, як двоє людей, які разом насолоджуються рідкісним незалежним фільмом. Щоб взяти це до уваги, просто трансформується кожен голос, зважуючи двох користувачів таким чином, що загально оцінюваним пунктам надається менше значення.
3) Підсилення випадку. Підхід дуже простий. Просто експоненціально підсилюється кожна вага показника, щоб вищі ваги ставали вищими, а менші - меншими. Як правило, це працює не дуже добре, але можна помітити незначне поліпшення.

Розглянувши головні характеристики алгоритмів колаборативної фільтрації, заснованих на пам’яті можна виділити декілька переваг:

1) Якість прогнозування досить хороша.
2) Алгоритм колаборативної фільтрації відносно простий у реалізації у будь-якій системі.
3) Оновлення бази даних досить легке, адже алгоритм використовує всю базу даних кожного разу, коли робить прогноз.

Також, можна виділити і декілька недоліків алгоритмів такого типу:

1) Оскільки алгоритм використовує для аналізу всю базу даних коли робить прогноз, то база повинна бути постійно у пам’яті, що сповільнює роботу.
2) Іноді така система не може зробити вірну рекомендацію, або рекомендацію взагалі. Це може статися через те, що у активного користувача немає спільних елементів з усіма людьми, які оцінили цільовий елемент.
3) Перекриває дані. Вся випадкова мінливість оцінок людей сприймається як причинно-наслідковий зв'язок, що може бути справжньою проблемою. Іншими словами, алгоритми, засновані на пам'яті, взагалі не узагальнюють дані.

1.2.2 Алгоритми засновані на моделі

Алгоритми, засновані на даних, базуються на моделях, які надають рекомендації щодо товарів, спершу розробляючи модель оцінок користувачів. Алгоритми цієї категорії мають імовірнісний підхід і представляють процес колаборативної фільтрації, як обчислення очікуваного значення прогнозування користувача, враховуючи його оцінки інших елементів [3].

Процес побудови моделі виконується за допомогою різних алгоритмів машинного навчання, таких як байєсівська мережа, кластеризація та підходи, засновані на правилах.

Модель мережі Байєса формулює імовірнісну модель для задачі колаборативної фільтрації. Кластеризаційна модель сприймає колаборативну фільтрацію, як задачу класифікації і працює шляхом кластеризації схожих користувачів в одному класі та оцінки ймовірності того, що конкретний користувач перебуває в певному класі C, і звідти обчислює умовну ймовірність оцінок. Підхід, заснований на правилах, застосовує алгоритми виявлення правил асоціації до асоціації між товарами, що були придбані спільно, а потім генерує рекомендації щодо товарів на основі міцності зв'язку між предметами.

Основна ідея систем рекомендацій на основі пам’яті полягає в тому, щоб розрахувати та використовувати схожість між користувачами або елементами та використовувати їх як «ваги» для прогнозування рейтингу для користувача та елемента. Ту саму ідею можна використовувати в алгоритмах, заснованих на моделях: подібність між користувачами та/або елементами можна обчислити, а потім зберегти як модель, а потім можна використовувати збережені значення подібності для прогнозування оцінок. Ці моделі також можна будувати, використовуючи схожість між елементами, а не користувачами, і іноді це доцільніше робити. Наприклад, NetflixPrize дані містять трохи менше 5 000 000 користувачів, але лише трохи більше 17 000 фільмів. Це робить можливим те, що отримана модель для елементів буде меншою, ніж для користувачів.

Така система, що базується на моделі, також часто дозволяє обрізати модель, щоб зробити систему більш масштабованою. Зокрема, ми можемо обмежити кількість подібних сутностей (користувачів або предметів), які ми зберігаємо для кожного об’єкта. Іншими словами, ми зберігаємо лише k найбільш подібних сутностей. Дослідники виявили, що зберігання обмеженої кількості подібних об’єктів часто мало впливає на точність прогнозів.

Серед переваг таких алгоритмів колаборативної фільтрації можна виділити:

1) Масштабованість. Більшість моделей, отриманих на основі модельних алгоритмів, набагато менші, ніж фактичний набір даних, так що навіть для дуже великих наборів даних модель виявляється досить малою для ефективного використання. Це надає масштабованості загальній системі.
2) Швидкість прогнозування. Системи на основі моделі також, швидше за все, будуть швидшими , принаймні в порівнянні з системами на основі пам'яті, оскільки час, необхідний для запиту моделі (на відміну від цілого набору даних), як правило, набагато менший, ніж час, необхідний для запиту весь набір даних.
3) Уникнення переобладнання. Якщо набір даних, над яким збудована модель, є достатньо репрезентативним для реальних даних, легше спробувати уникнути переобладнання системами, заснованими на моделях.

Окрім переваг, також існує пара недоліків:

1) Негнучкість. Оскільки побудова моделі часто є трудомістким процесом, як правило, складніше додавати дані до систем на основі моделей, роблячи їх негнучкими.
2) Якість прогнозів. Так як не використовується вся доступна інформація (весь набір даних), можливо, що в системах на основі моделей не отримуються настільки точні передбачення, як у системах на основі пам'яті. Однак слід зазначити , що якість прогнозів залежить від способу побудови моделі.

1.3 Алгоритм колаборативної фільтрації заснований на елементах

Підхід, що ґрунтується на елементах, розглядає набір елементів, які оцінив цільовий користувач, і обчислює, наскільки вони схожі на цільовий елемент i, а потім вибирає k найбільш подібні предмети $\{i_1, i_2, \dots, i_k\}$. У той же час їх відповідні подібності $\{s_{i1}, s_{i2}, \dots, s_{ik}\}$ також обчислюються. Після того, як знайдені найбільш подібні предмети, прогноз потім обчислюється шляхом взяття середньозваженої цілі оцінки користувачів щодо подібних предметів.

Один із найважливіших кроків у колаборативній фільтрації на основі товарів полягає в обчисленні подібності між елементами та потім обираються найбільш схожі елементи. Основна ідея в обчисленні подібності між двома елементами i та j полягає у ізолюванні користувачів, які оцінили обидва ці елементи, а потім застосовувати методику обчислення подібності для визначення подібність $s_{i,j}$. Рисунок 1.2 ілюструє цей процес. Тут рядки матриці представляють користувачів, а стовпці - предметів.

Існує ряд різних способів обчислення подібності між предметами. Це подібність на основі косинусів, подібність на основі кореляції та подібність на основі скоригованих косинусів [1].

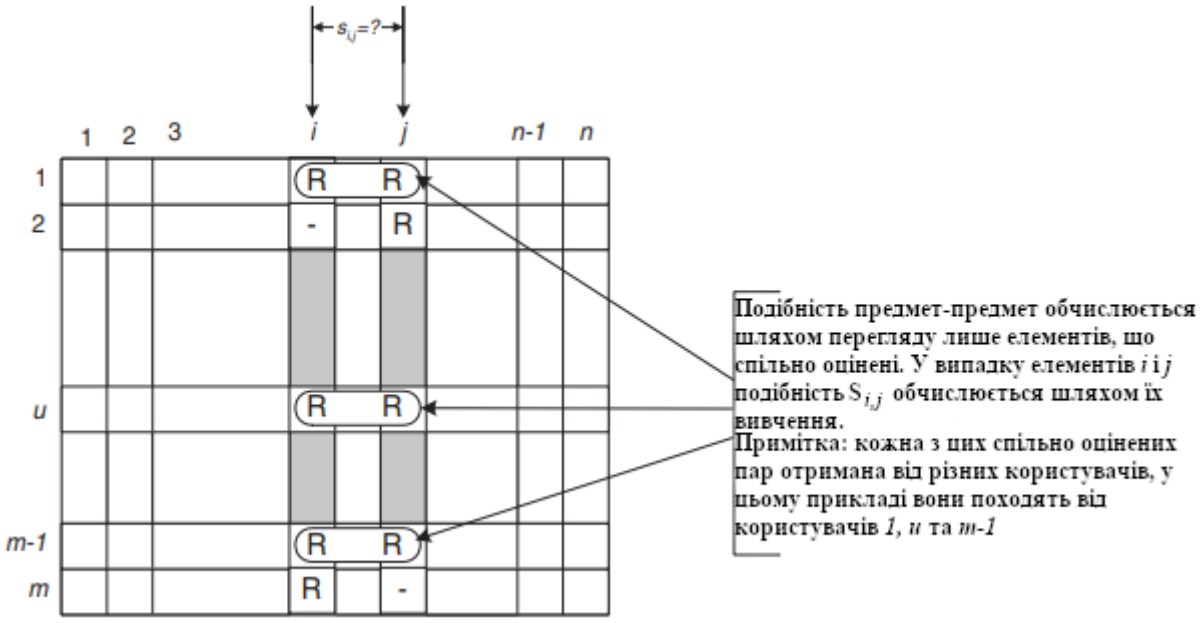

Рис. 1.2 Ізоляція спільно оцінених елементів та розрахунок схожості

1.3.1 Подібність на основі косинусів

У цьому випадку два елементи розглядаються як два вектори в m-мірний простір користувача. Подібність між ними вимірюється обчисленням косинуса кута між цими двома векторами. Формально в m x n рейтинговій матриці
на Рисунку 1.2, подібність між елементами i та j, позначене як
$sim(i,j)$ задано як

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2},$$

де «.» позначає точковий добуток двох векторів [1].

1.3.2 Подібність на основі кореляції

У цьому методі подібність між двома елементами і та j вимірюється обчисленням кореляційних корекцій Пірсона $corr_{i,j}$. Щоб зробити коректне обчислення кореляційним, потрібно спочатку відокремити випадки, що оцінюються спільно (тобто випадки, коли користувачі оцінювали як i, так і j), як показано на Рисунку 1.2. Нехай набір користувачів, які обидва оцінені i та j позначаються U, тоді кореляційна схожість виглядає так

$$sim(i,j) = \frac{\sum_{u \in U}\left(R_{u,i} - \bar{R}_i\right)\left(R_{u,j} - \bar{R}_j\right)}{\sqrt{\sum_{u \in U}\left(R_{u,i} - \bar{R}_\iota\right)^2} * \sqrt{\sum_{u \in U}\left(R_{u,j} - \bar{R}_j\right)^2}}$$

Тут $R_{u,i}$ позначає рейтинг користувача u елемента i, $\bar{R}_\iota$ - середня оцінка i-го елемента [1].

1.3.3 Подібність на основі скоригованих косинусів

Однією з принципових відмінностей між обчисленнями подібності в колаборативній фільтрації на основі користувача та колаборативній фільтрації на основі елементів є те, що у випадку для фільтрацій, заснованих на користувачеві, подібність обчислюється по рядках матриці, але у випадку колаборативної фільтрації, що базується на елементах, подібність обчислюється вздовж стовпців, тобто кожна пара в наборі, що оцінюється, відповідає різному користувачеві (Рисунок 1.2). Обчислення подібності за допомогою базової косинусної міри у випадку фільтрації на основі елементів має один важливий недолік/відмінність в шкалі оцінок між різними користувачами не враховуються. Відрегульована схожість косинусів компенсує цей недолік, віднімаючи відповідне середнє значення для кожного з загальних рейтингів. Формально подібність між елементами і та j використовує цю формулу

$$sim(i,j) = \frac{\sum_{u \in U}\left(R_{u,i} - \bar{R}_u\right)\left(R_{u,j} - \overline{R_u}\right)}{\sqrt{\sum_{u \in U}\left(R_{u,i} - \overline{R_u}\right)^2} * \sqrt{\sum_{u \in U}\left(R_{u,j} - \overline{R_u}\right)^2}}.$$

Тут $\overline{R_u}$ - середнє значення u-го користувача [1].

1.3.4 Розрахунок прогнозу

Найважливішим кроком у системі колаборативної фільтрації є створення вихідного інтерфейсу з точки зору прогнозування. Як тільки виділиться безліч найподібніших елементів на основі вимірів подібності, наступним кроком є вивчення рейтингів цільових користувачів та використання техніки отримання прогнозів.

Можна виділити дві таких техніки:

1) Зважена сума. Як випливає з назви, цей метод обчислює передбачення за елементом i для користувача u шляхом обчислення суми оцінок користувача щодо предметів, подібних до i. Кожна оцінка зважується відповідною подібністю $s_{i,j}$ між елементами i та j. Формально, використовуючи поняття, наведене на Рисунку 1.3, можна позначати передбачення $P_{u,i}$ як

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N}(S_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N}(|s_{i,N}|)}.$$

В основному, такий підхід намагається визначити, як активний користувач оцінює подібні елементи. Зважена сума масштабується на сумою термінів подібності, щоб переконатись, що передбачення знаходиться в заданому діапазоні.

2) Регресія. Цей підхід подібний до методу зваженої суми, але замість безпосереднього використання рейтингів подібних елементів він використовує наближення рейтингів на основі регресійної моделі. На практиці подібність, обчислена за допомогою косинусів або вимірювань співвідношення, може ввести в оману в тому сенсі, що два рейтингові вектори можуть бути віддаленими (в евклідовому розумінні), але можуть мають дуже високу схожість. У такому випадку використовуються необроблені рейтинги так званого «подібного» елемента може призвести до поганого прогнозування. Основна ідея - використовувати ту саму формулу, що і техніка зваженої суми, але замість того, щоб використовувати аналогічні значення N вихідних рейтингових значень $R_{u,N}$, ця модель використовує їх наближені значення $R_{u,N}^{\acute{}}$ на основі моделі лінійної регресії. Якщо ми позначимо відповідні вектори цільового елемента i та подібного елемента N через R_i та R_N, модель лінійної регресії може бути виражена як

$$\overline{R_N^{\acute{}}} = \alpha \overline{R}_\iota + \beta + \epsilon$$

Параметри α та β моделі регресії визначаються шляхом перегляду обох векторів рейтингу. ϵ - помилкою регресійної моделі [1].

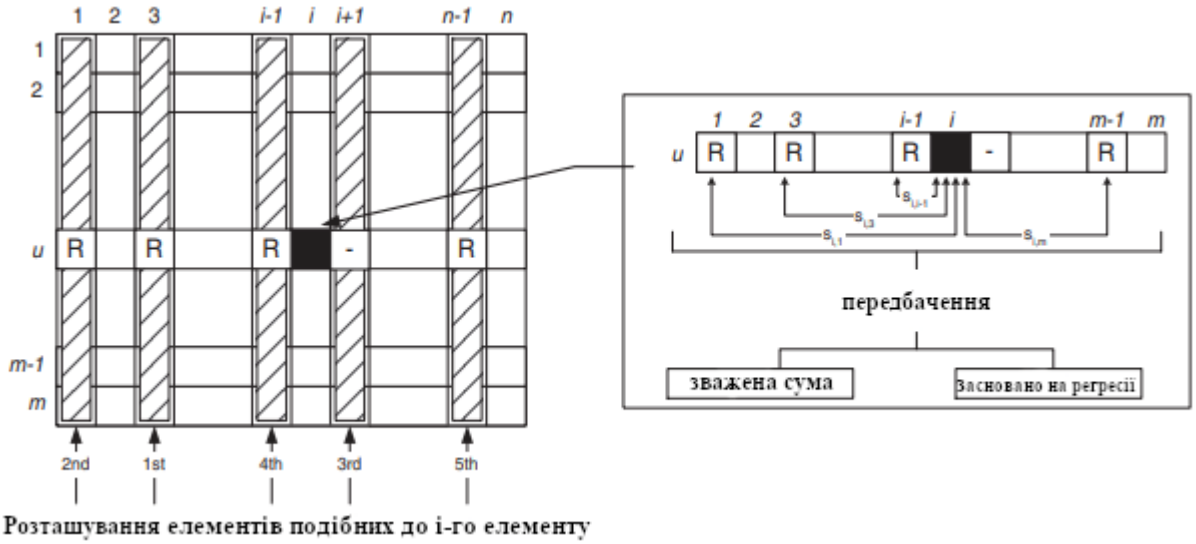

Рис. 1.3 Алгоритм, заснований на елементах. Генерація передбачення показана для 5 сусідів

1.4 Нейронна мережа

Штучна нейронна мережа створена за зразком біологічної нейронної мережі. Як і біологічна нейронна мережа, ANN є взаємозв'язком вузлів, аналогічним нейронам. Кожна нейронна мережа має три критичні компоненти: характер вузла, топологію мережі та правила навчання. Символ вузла визначає спосіб обробки сигналів вузлом, наприклад кількість входів і виходів, пов'язаних з вузлом, вагу, пов'язану з кожним входом і виходом, та функцію активації. Топологія мережі визначає способи організації та підключення вузлів. Правила навчання визначають спосіб ініціалізації та коригування ваг.

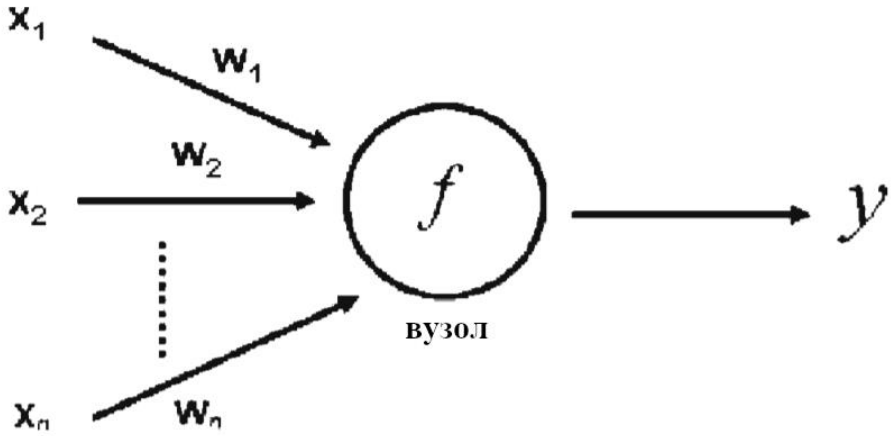

Рис. 1.4 Базова модель вузла: x_i= вхід, w_i= вага, f= передавальна функція, y_i= вихід

Основна модель вузла в ANN наведена на Рисунку 1.4. Кожен вузол отримує кілька входів від інших через з'єднання, що мають пов'язані ваги, аналогічні силі синапсу. Коли зважена сума входів перевищує порогове значення вузла, він активує і передає сигнал через передавальну функцію і передає його сусіднім вузлам. Цей процес можна виразити як математичну модель:

$$y = f\left(\sum_{i=0}^{n} w_i x_i - T\right),$$

де y - вихід вузла, f - передавальна функція, w_i - вага вхідного x_i, T - порогове значення. Функція передачі має багато форм. Нелінійна передавальна функція є більш корисною, ніж лінійна, оскільки лише декілька задач є лінійно відокремлюваними. Найпростішою є ступінчаста функція (див. Рис. 1.5):

$$y = \begin{cases} 0 \dots if \sum_{i=0}^{n} w_i x_i > T \\ 1 \dots if \sum_{i=0}^{n} w_i x_i < T \end{cases}$$

Сигмоїдна функція також часто використовується як функція активації, оскільки функція та її похідна є безперервними (див. Рис. 1.5):

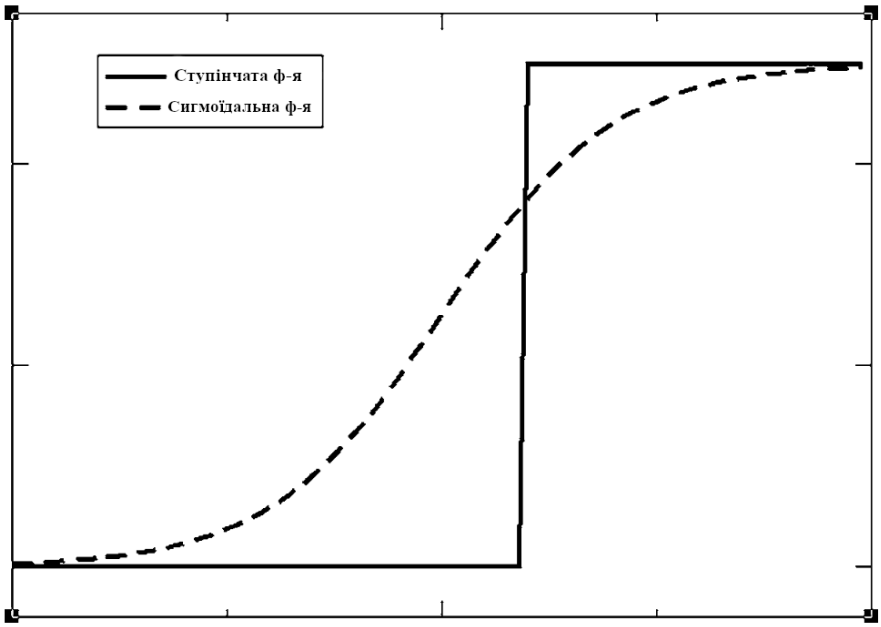

Рис. 1.5 Передавальна функція

На Рис. 1.6 показано загальну архітектуру ANN. Вузли організовані в лінійні масиви, які називаються шарами. Зазвичай є вхідні шари, вихідні шари та приховані шари. Не може бути жодного до декількох прихованих шарів. Проектування топології мережі передбачає визначення кількості вузлів на кожному шарі, кількості шарів у мережі та шляху з'єднань між вузлами. Зазвичай ці фактори спочатку встановлюються за допомогою інтуїції та оптимізуються за допомогою численних циклів експериментів. Також деякі раціональні методи можуть бути використані для проектування нейронної мережі. Наприклад, генетична нейронна мережа (GNN) використовує загальний алгоритм для вибору вхідних функцій для нейронної мережі.

Існує два типи зв’язку між вузлами. Одне - одностороннє з'єднання без зворотного циклу. Інший - це зворотне з'єднання, в якому вихід вузлів може бути входом до попередніх або однакових рівнів. На основі вищезгаданого типу з'єднань нейронні мережі можна класифікувати на два типи: мережа прямого зв'язку та мережа зворотного зв'язку, як показано на Рис. 1.7 та 1.8. Оскільки сигнал рухається лише в один бік, мережа прямого пересилання є статичною, тобто один вхід асоціюється з одним конкретним виходом. Мережа зворотного зв'язку є динамічною. Для одного входу стан мережі зворотного зв'язку змінюється протягом багатьох циклів, поки не досягне точки рівноваги, тому один вхід виробляє серію виходів. Перцептрон - це широко використовувана мережа зворотного зв'язку. Деякі відомі мережі зворотного зв'язку включають мережу Хопфілда та карти самоорганізації Кохонена [4].

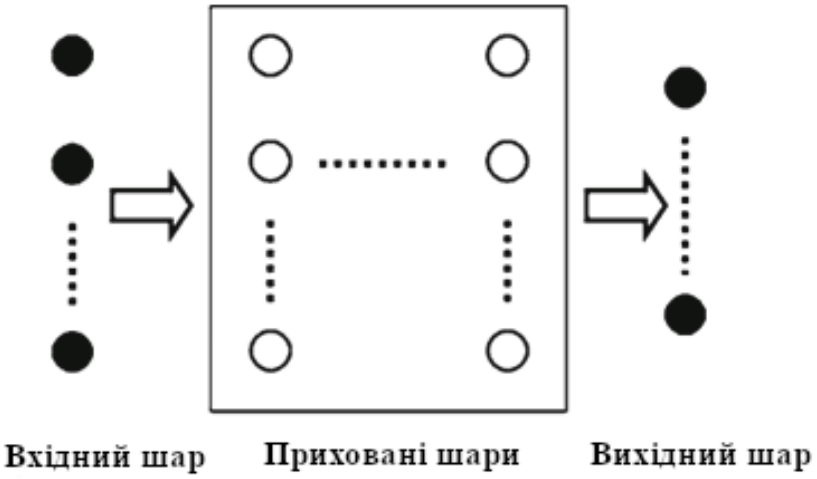

Рис. 1.6 Загальна архітектура ANN

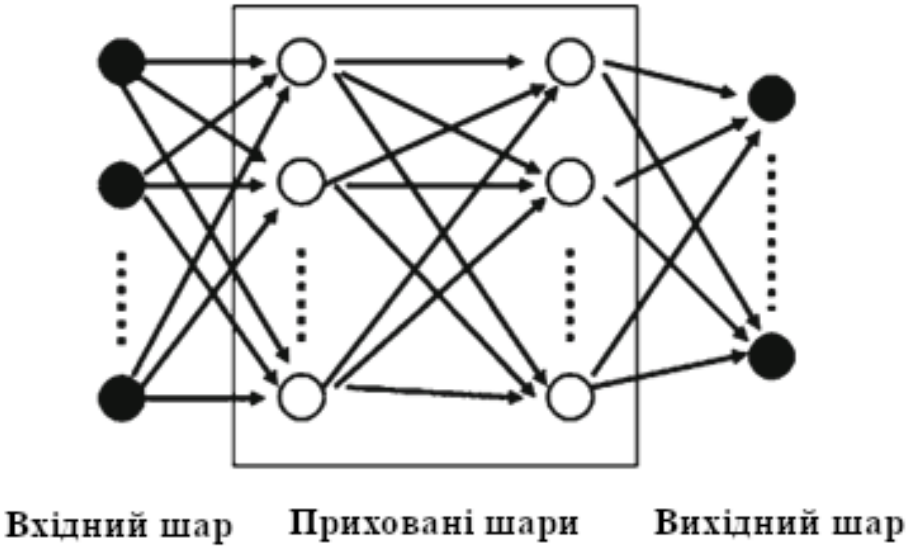

Рис. 1.7 Мережа прямого зв'язку

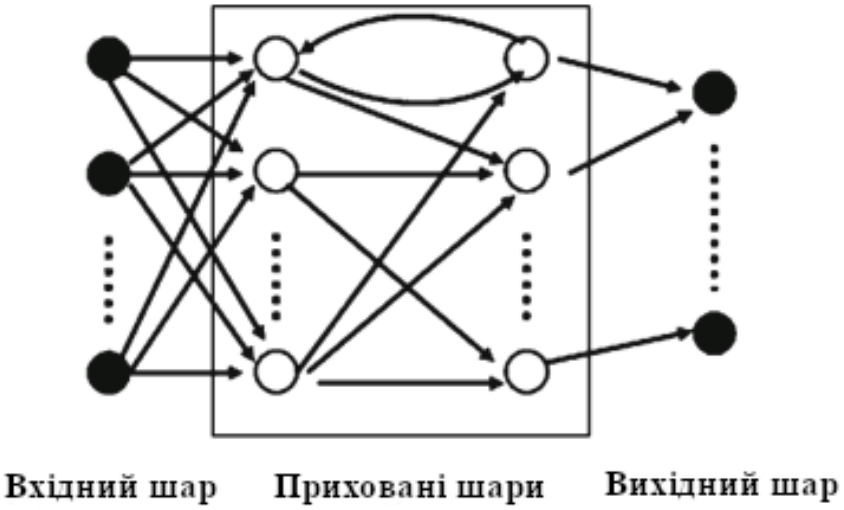

Рис. 1.8 Мережа зворотного зв'язку

1.5 Згорткова нейронна мережа

Згорткова нейронна мережа (ЗНМ) - це модель глибокого навчання для обробки даних, яка має сітчасту структуру, таку як зображення, яка натхнена організацією зорової кори тварин і призначена для автоматичного та адаптивного вивчення просторових ієрархій особливостей, від моделей низького до високого рівня. ЗНМ - це математична конструкція, яка, як правило, складається з трьох типів шарів (або будівельних блоків): згортки, об'єднання та повністю пов'язаних шарів. Перші два, згортання та об’єднання шарів, виконують вилучення ознак, тоді як третій, повністю зв’язаний шар, відображає вилучені об’єкти у кінцевий результат, такий як класифікація. Рівень згортки відіграє ключову роль у ЗНМ, який складається з набору математичних операцій, таких як згортка - спеціалізований тип лінійних операцій. У цифрових зображеннях значення пікселів зберігаються у двовимірній (2D) сітці, тобто масиві чисел (Рис.1.9) і застосовується невелика сітка параметрів, яка називається ядром, оптимізованим екстрактором функцій у кожному положенні зображення, що робить ЗНМ високоефективними для обробки зображень, оскільки функція може виникати де завгодно на зображенні. По мірі того, як один шар подає свої результати в наступний шар, витягнуті функції можуть ієрархічно і поступово

ускладнюватися. Процес оптимізації таких параметрів, як ядра, називається навчанням, яке виконується таким чином, щоб мінімізувати різницю між виходами та основними мітками істини за допомогою алгоритму оптимізації, названого зворотнім розповсюдженням та зниженням градієнта, серед іншого.

Архітектура ЗНМ включає декілька будівельних блоків, таких як згорткові шари, об'єднуючі шари та повністю пов'язані шари. Типова архітектура складається з повторень стека з декількох шарів згортки та шару об'єднання, за яким слідує один або кілька повністю з'єднаних шарів. Крок, де вхідні дані перетворюються у вихідні через ці шари, називається прямим поширенням. Незважаючи на те, що операції згортки та об'єднання стосуються 2D-CNN, подібні операції можуть виконуватися і для тривимірних (3D) -CNN [5].

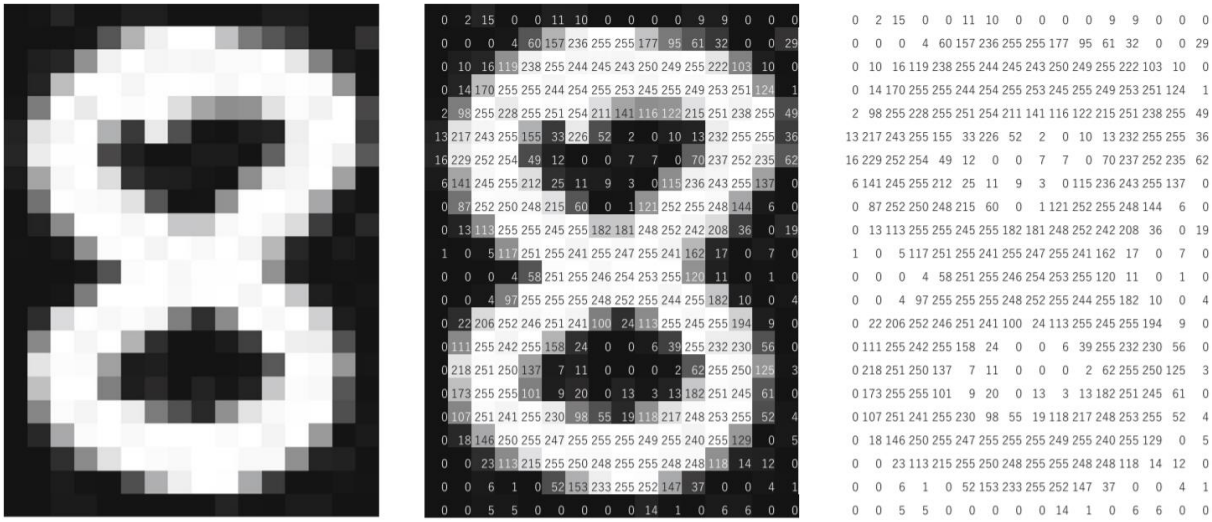

Рис. 1.9 Комп'ютер бачить зображення у вигляді масиву чисел. Матриця справа складається з чисел, що показують яскравість пікселів[5]

1.5.1 Згортковий шар

Шар згортки є фундаментальним компонентом архітектури ЗНМ, який виконує виділення ознак, який зазвичай складається з комбінації лінійних та нелінійних операцій, тобто, операція згортки та функція активації.

Згортка - це спеціалізований тип лінійної операції, що використовується для вилучення ознак, де на вхід подається невеликий масив чисел, який називається

ядром, що є масивом чисел, який називається тензором. Поелементно добуток між кожним елементом ядра та вхідним тензором обчислюється в кожному розташуванні тензора і підсумовується для отримання вихідного значення у відповідному положенні вихідного тензора, що називається картою ознак (Рис.1.10). Ця процедура повторюється із застосуванням декількох ядер для формування довільної кількості карт функцій, які представляють різні характеристики вхідних тензорів. Отже, різні ядра можна розглядати як різні екстрактори особливостей. Два ключових гіперпараметри, що визначають операцію згортки - це розмір та кількість ядер. Перший, як правило, становить 3 × 3, але іноді 5 × 5 або 7 × 7. Останній є довільним і визначає глибину вихідних характеристичних карт [5].

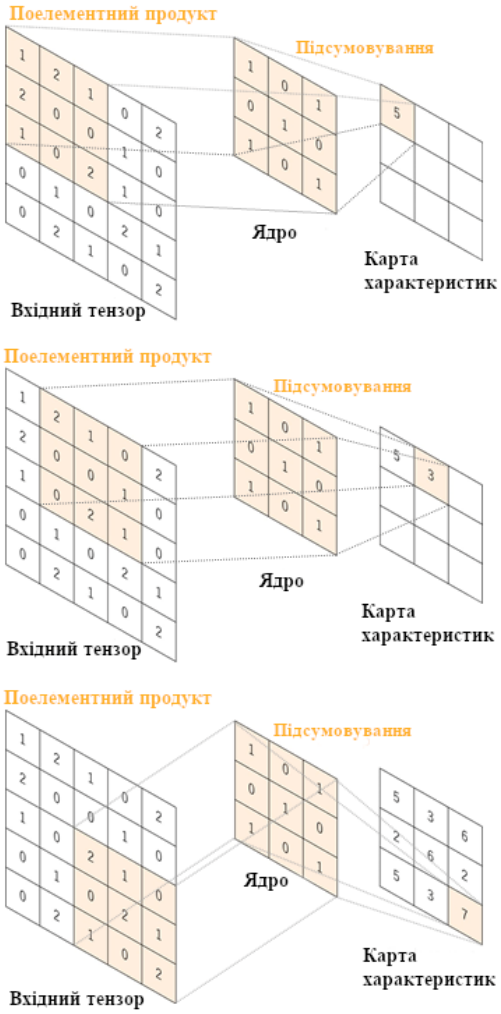

Рис. 1.10 Приклад операції згортки з розміром ядра 3 × 3, без заповнення та кроком 1

Описана вище операція згортки не дозволяє центру кожного ядра перекривати крайній елемент вхідного тензора і зменшує висоту та ширину вихідної карти функцій порівняно з вхідним тензором. Доповнення, як правило, нульове заповнення, - це техніка вирішення цієї проблеми, коли рядки та стовпці нулів додаються з кожного боку вхідного тензора, щоб відповідати центру ядра на самому зовнішньому елементі і зберігати однакову площину розмірність за допомогою операції згортки (Рис. 1.11). Сучасні архітектури CNN зазвичай використовують нульове заповнення, щоб зберегти площинні розміри, щоб нанести більше шарів. Без нульового заповнення кожна наступна карта об’єктів буде зменшуватися після операції згортки [5].

Відстань між двома послідовними положеннями ядра називається кроком, який також визначає операцію згортки. Загальним вибором кроку є 1, однак іноді використовується крок, більший за 1, для того, щоб домогтися дискретизації функціональних карт. Альтернативною технікою для зменшення вибірки є операція об’єднання, як описано нижче.

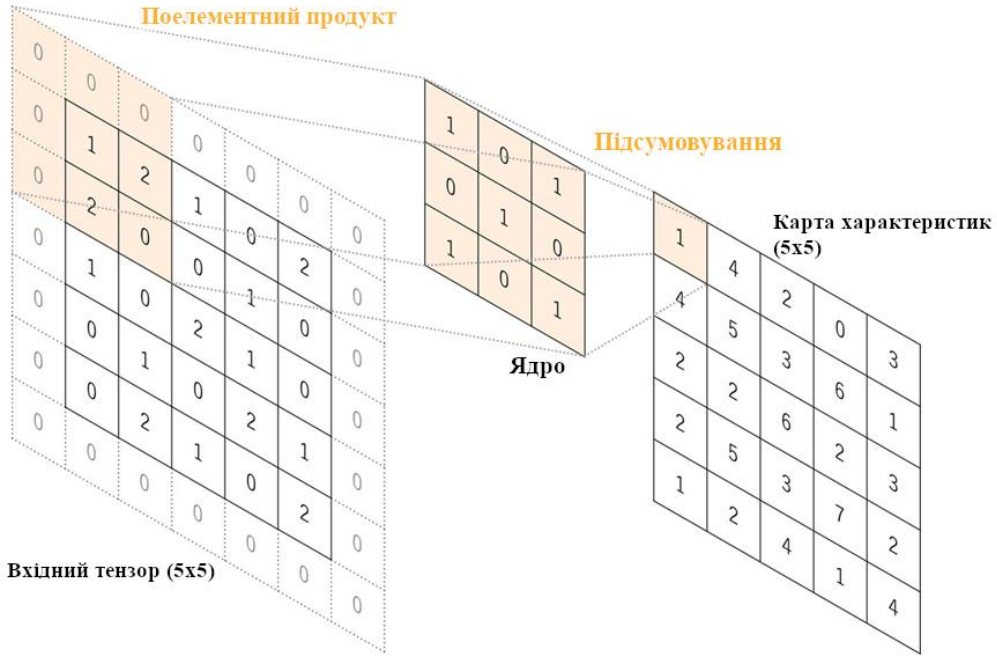

Рис. 1.11 Операція згортки з нульовими відступами, щоб зберегти розміри площини.

Ключовою особливістю операції згортки є розподіл ваги: ядра розподіляються по всіх позиціях зображення. Спільне використання ваги створює наступні характеристики операцій згортки:

1) Надання локальним шаблонам ознак, вилучених перекладом ядер b, інваріантними, коли ядра пересуваються по всіх позиціях зображення та виявляють вивчені локальні шаблони.
2) Вивчення просторових ієрархій шаблонів об’єктів шляхом зменшення вибірки в у поєднанні з операцією об'єднання, що призводить до захоплення дедалі більшого поля зору.
3) Підвищення ефективності моделі за рахунок зменшення кількості параметрів, які потрібно засвоїти в порівнянні з повністю підключеними нейронними мережами.

Процес навчання моделі CNN щодо рівня згортки полягає у визначенні ядер, які найкраще працюють для даного завдання на основі заданого набору даних навчання. Ядра - це єдині параметри, які автоматично засвоюються під час тренувального процесу в шарі згортки. З іншого боку, розмір ядер, кількість ядер, прокладки та крок - це гіперпараметри, які потрібно встановити перед початком навчального процесу.

Потім виходи лінійної операції, такі як згортка, передаються через нелінійну функцію активації. Хоча раніше використовувались гладкі нелінійні функції, такі як сигмоїдна або гіперболічна дотична $(tanch)$, оскільки вони є математичними зображеннями поведінки біологічних нейронів, найпоширенішою функцією нелінійної активації, що використовується в даний час, є випрямлена лінійна одиниця (ReLU), яка просто обчислює функція: $f(x) = \max(0, x)$.

1.5.2 Шар пулінгу

Шар об'єднання забезпечує типову операцію зменшення вибірки, яка зменшує розмірність площини карт об'єктів, щоб внести незмінність перекладу до невеликих зсувів та спотворень, а також зменшити кількість наступних параметрів, що

піддаються вивченню. Слід зазначити, що в жодному з шарів об’єднання немає параметру, що навчається, тоді як розмір фільтра, крок та відступ - це гіперпараметри в операціях об’єднання, подібно до операцій згортання.

Найпопулярнішою формою операції об'єднання є максимальне об'єднання, яке витягує патчі з вхідних карт функцій, виводить максимальне значення в кожному патчі та відкидає всі інші значення (Рис. 1.12). На практиці зазвичай використовується максимальне об’єднання з фільтром розміром 2×2 з кроком 2. Це зменшує площину розмірності карт об'єктів у 2 рази на відміну від висоти та ширини, розмір глибини карт об'єктів залишається незмінним [5].

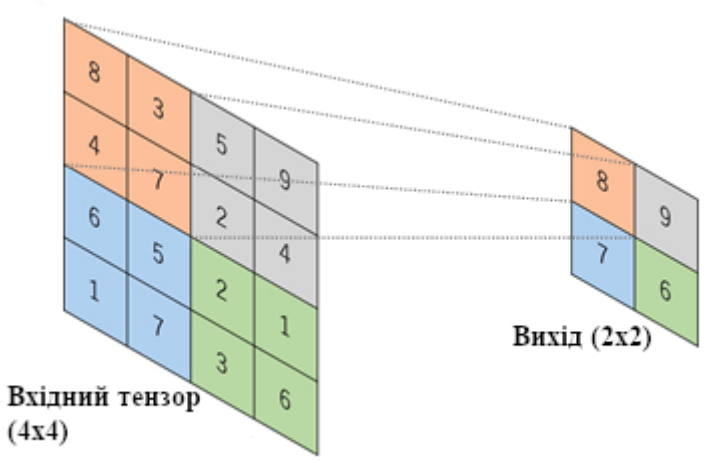

Рис. 1.12 Приклад операції максимального об’єднання з розміром фільтра 2 × 2, без заповнення та кроком 2

Ще однією операцією об’єднання, на яку слід звернути увагу, є загальне середнє об’єднань. Глобальне середнє об’єднання здійснює екстремальний тип зменшення дискретизації, коли карта об’єктів із розміром висоти × ширини зменшується в масив 1×1, просто беручи середнє значення всіх елементів на кожній карті об’єктів, тоді як глибина карт об’єктів зберігається. Ця операція, як правило, застосовується лише один раз перед повнозв'язними шарами. Переваги застосування загального середнього пулінгу полягають у наступному: зменшує кількість засвоюваних параметрів і дозволяє ЗНМ приймати входи змінного розміру.

1.5.3 Повнозв'язний шар

Карти вихідних ознак остаточної згортки або шару об'єднання зазвичай сплющуються, тобто перетворюються в одновимірний (1D) масив чисел (або вектор) і з'єднуються з одним або декількома повністю з'єднаними шарами, також відомими як щільні шари, в якому кожен вхід пов’язаний з кожним виходом вагою, який можна дізнатись. Після створення об’єктів, вилучених шарами згортки та зменшених вибіркою шарів пулінгу (об’єднання), вони відображаються підмножиною повністю зв’язаних шарів із кінцевими виходами мережі, такими як ймовірності для кожного класу в задачах класифікації. Кінцевий повністю зв’язаний шар зазвичай має таку ж кількість вихідних вузлів, як і кількість класів. За кожним повністю зв’язаним шаром слідує нелінійна функція, така як ReLU та інші [5].

1.5.4 Функція активації останнього шару

Функція активації, застосована до останнього повністю підключеного шару, зазвичай відрізняється від інших. Відповідно до кожного завдання потрібно вибрати відповідну функцію активації. Функція активації, застосована до багатокласової класифікаційної задачі - це функція softmax, яка нормалізує вихідні реальні значення з останнього повністю підключеного шару до імовірностей цільового класу, де кожне значення коливається від 0 до 1, а всі значення складають до 1. Типовий вибір останнього шару функція активації для різних типів завдань наведено в Табл. 1.1 [5].

Таблиця 1.1

Список найпопулярніших функцій активації для різних задач

Задача	Функція активації останнього шару
Бінарна класифікація	Сигмоїдальна
Багатокласова однокласна класифікація	Softmax
Багатокласова багатокласна класифікація	Сигмоїдальна
Регресія до безперервних значень	Тотожна

ВИСНОВКИ ДО РОЗДІЛУ 1

Нейромережі, а особливо згорткові, являються сучасною та перспективною технологією, яка вже активно використовується у сьогоденні. Згорткові нейромережі набули популярності через стійкість алгоритму до модифікацій зображення, узагальнення характеристик зображення, та значне прискорення у обчисленнях, через можливість розпаралелювання. Всі ці характеристики дають перевагу у якості та швидкості аналізу зображень.

Основною проблемою більшості наявних підходів у рекомендаційних системах являється проблема «холодного старту», тобто недостатня кількість вхідних даних для повноцінної роботи алгоритмів рекомендаційних систем. Отже, для вирішення даної проблеми потрібно реалізувати алгоритм аналізу графічних зображень на базі згорткової нейромережі.

2 РОЗДІЛ
ВИБІР ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ ТА РОЗРОБКА АЛГОРИТМУ

2.1 Вибір інструментів для розробки

На даний час можна виділити безліч інструментів для розробки програмного забезпечення. Це стосується мов програмування, середовищ для розробки, баз даних, додаткових фреймворків для реалізації певних задач та інше.

Сьогодні налічується більше 265 мов програмування і кожна з них має своє призначення та особливості. З огляду на рейтинги, можна зробити висновки, що найпопулярнішими є Java, Javascript, Python, C++ та C#. Мова програмування повинна бути досить багатофункціональною, гнучкою, надійною, кросплатформеною та відносно простою у використанні. Під ці характеристики підпадає мова програмування Java, тому була обрана для розробки програмного забезпечення у цій роботі.

До найпопулярніших середовищ для розробки (IDE) на мова Java можна віднести Eclipse, NetBeans та IntelijIdea.

IntelijIdea – середовище для розробки програмного забезпечення, яке написано на мові Java для розробки на Java. Це середовище забезпечує певні функції, такі як автодоповнення коду шляхом аналізу контексту, навігація по кодовій базі проекту, яка дозволяє переходити до декларацій методів безпосередньо. Також IntelijIdea має широкий вибір інструментів для рефакторингу коду, його налагодження та середовище само може запропонувати змінити частину коду з відповідними підказками [6].

Це середовище розробки має вбудовану інтеграцію з системами контролю версій Git, Mercurial, SVN та Perforce. Існує можливість роботи з базами даних SQL Server, MySQL, PostgreSQL та іншими прямо з середовища розробки.

Середовище розробки IntelijIdea має підтримку великої кількості мов програмування. Звичайно, це Java та Kotlin, які безпосередньо настроєні з початку запуску IDE, а більше 15 інших мов доступні з допомогою плагінів.

IntelijIdea має дві версії: «Ultimate» (платна) та «Community» (безкоштовна).

Eclipse - це інтегроване середовище розробки (IDE), що використовується в комп'ютерному програмуванні. Воно містить базовий робочий простір та розширювану систему плагінів для налаштування середовища. Eclipse написаний здебільшого на Java, і його основне використання - для розробки програм Java, але він також може використовуватися для розробки додатків на інших мовах програмування за допомогою плагінів, включаючи Ada , ABAP , C , C ++ , C # та інші [7].

Комплект розробки програмного забезпечення Eclipse (SDK) - це безкоштовне програмне забезпечення з відкритим кодом, випущене на умовах Загальної ліцензії Eclipse.

NetBeans - це інтегроване середовище розробки з відкритим кодом. IDE NetBeans підтримує розробку всіх типів програм Java (Java SE (включаючи JavaFX), Java ME, WEB, EJB та мобільні додатки). Серед інших можливостей - система проектів на основі Ant, підтримка Maven, рефакторинг, контроль версій (підтримка CVS , Subversion , Git , Mercurial та Clearcase). NetBeans дозволяє розробляти додатки із набору модульних програмних компонентів, які називаються модулями. Кожен модуль забезпечує чітко визначені функції, такі як підтримка мови Java, редагування або підтримка системи версій CVS та SVN. NetBeans містить всі модулі, необхідні для розробки Java, за одне завантаження, що дозволяє користувачеві негайно розпочати роботу. Модулі також дозволяють розширити NetBeans. Нові функції, такі як підтримка інших мов програмування, можна додати, встановивши додаткові модулі [8].

З огляду на найпопулярніші середовища розробки для мови програмування Java, можна виділити середовище IntelijIdea. Це IDE дозволяє швидко та продуктивно розробляти програмний продукт, так як має можливість автодоповнення коду, зв’язку з базою даних безпосередньо із середовища та має широкий спектр допоміжних інструментів для редагування, заміни коду.

2.2 Вибір допоміжних технологій для розробки

Для реалізації задачі по розробці рекомендаційної системи з допомогою технології нейромереж знадобиться використання бази даних, фреймворк для роботи з нейромережами та інше. Для того, щоб продемонструвати роботу рекомендаційної системи, потрібно розробити WEB-додаток.

Існує два підходи до створення WEB-додатків на мові Java. Один з них – Java EE (Java Enterprise Edition). Цей підхід доволі складний у реалізації та дуже затратний по часу, тому на заміну йому у всьому світі використовують фреймворк Spring (Spring Boot).

Spring (Spring Boot) - фреймворк і інверсія контролю контейнера для платформи Java. Основні функції фреймворку можуть використовувати будь-які додатки Java. Spring Framework складається з функцій, організованих приблизно в 20 модулів. Ці модулі згруповані як показано на Рис. 2.1 [9]:

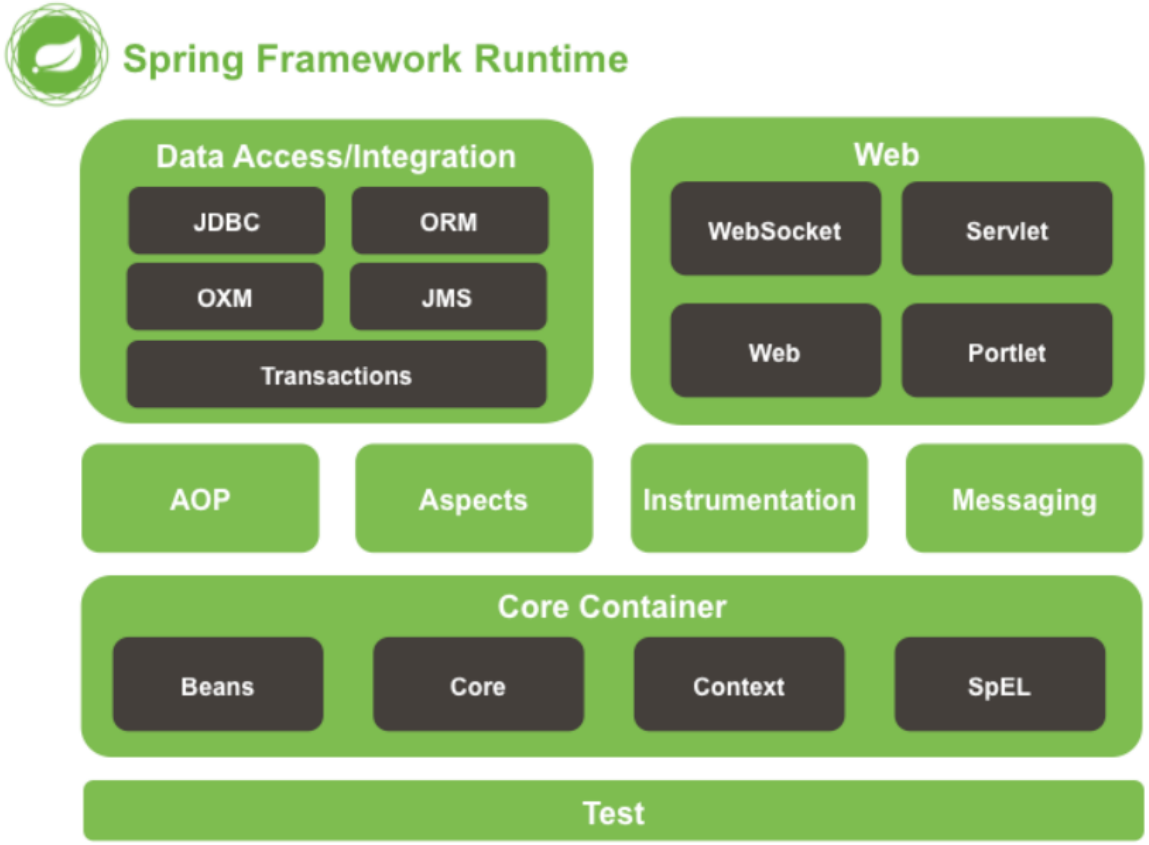

Рис. 2.1 Структура фреймворка Spring[10]

Наявність модулів дозволяє досить гнучко підходити до розробки програмного забезпечення. Можна підключати тільки ті модулі, які будуть дійсно потрібні під час розробки.

Наприклад, якщо передбачена робота з базами даних, то можна підключити Spring Data JPA – це додатковий модуль, який дозволяє взаємодіяти з «сутностями», відображенням яких є таблиці у базі даних, структурування їх та обробка. Головною перевагою є те, що більшість операцій, що часто використовуються, вже визначені та не потребують додаткового налаштування.

Для того, щоб ще спростити розробку додатків, розробники Spring створили «фасад» над звичайними фреймворком Spring та назвали його Spring Boot. Spring Boot – це проект на рівні виконання (IO Execution) IO Spring Framework, який спрощує настроювання Spring та підключення до нього модулів.

Переваги SpringBoot:

1) Допомагає автоматично налаштувати всі компоненти для виробничого додатка Spring.
2) Допомагає уникнути всілякої ручної роботи з написання зразкового коду, анотацій та складних конфігурацій XML.
3) Поставляється із вбудованими HTTP-серверами, такими як Jetty та Tomcat, для тестування веб-додатків.
4) Скорочує час, витрачений на розробку.
5) Дозволяє легко підключатись до служб баз даних та черг, таких як Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ та багатьох інших.
6) Допомагає автоматично налаштувати всі компоненти для виробничого додатка Spring.
7) Полегшує настроювання залежності внутрішніх компонент і постачається з вбудованим контейнером для сервлетів.
8) Надає безліч плагінів, які розробники можуть використовувати для плавної та легкої роботи із вбудованими базами даних та вбудованими базами даних.

Для взаємодії з додатком потрібно визначити протокол спілкування. Існує два найпопулярніших типи REST та SOAP.

REST(REpresentational State Transfer) - призначений для використання переваг існуючих протоколів HTTP при використанні для WEB-API. Він дуже гнучкий, оскільки не прив’язаний до ресурсів чи методів і має можливість обробляти різні виклики та формати даних. Оскільки REST API не обмежений таким форматом XML, як SOAP, він може повертати кілька інших форматів залежно від того, що потрібно. Якщо служба дотримується цього стилю, вона вважається додатком «RESTful». REST дозволяє компонентам отримувати доступ та керувати функціями в іншій програмі.

REST API моделює взаємодію клієнт-сервер. Для спілкування потрібно відправляти відповідні запити. Запити мають різні типи («UPDATE», «POST», «GET», «DELETE») в залежності від мети. Наприклад, для отримання потрібен GET, а для збереження на сервері – POST.

Після того, як сервер отримує запит, відбувається його обробка. У залежності від того, чи успішною вона була повертається статус, який містить код. Код описує конкретну помилку, що сталася, або повідомлення про успішне виконання.

2.3 Вибір фреймворка для роботи з нейромережею

Для того, щоб реалізовувати певні задачі з використанням нейромереж на мові Java можна виділити кілька базових фреймворків. Основні це Neuroph та Deeplearning4j.

Neuroph представляє об’єктно-орієнтовану структуру штучної нейромережі, що написана на Java. Такий фреймворк використовується для того, щоб навчати програмно нейронні мережі на мові Java. Також цей фреймворк надає для роботи бібліотеку класів та інструмент з візуальним інтерфейсом GUI «easyNeurons» з функціями створення та навчання нейронних мереж.

Фреймворк Neuroph надає класи, що покривають основні потреби при роботі з нейромережами такі як штучний нейрон, нейронні зв’язки, вага нейрона, транспортна функція, правила навчання та інше. Фреймворк підтримує найпопулярніші архітектури нейронних мереж. Такими архітектурами є багатошаровий перцептрон з методом зворотного поширення, мережа Хопфілда та Кохонана. Базові класи фреймворка можна доповнювати та налаштовувати для того, щоб створювати особливі правила навчання та нейронні мережі. Також Neuroph підтримує можливість розпізнавання зображень та роботу з ними [11].

Фреймворк Deeplearning4j представляє собою бібліотеку, що надає можливості роботи з машинним навчанням та написана на мові Java і має досить широку обчислювальну базу алгоритмів машинного навчання. Deeplearning4j також надає можливість використовувати нестандартні архітектури нейронних мереж, такі як обмежена машина Больцмана, мережа переконань, автокодер з багатошаровим шумопоглинанням та рекурсивна мережа нейронних тензорів, word2vec, doc2vec і GloVe. Реалізація усіх архітектур можлива з підтримкою паралелізації та інтеграції з Apache Hadoop і Spark.

Deeplearning4j має інтеграцію з Tensorflow та Keras. Фреймворк має змогу імпортувати моделі з Tensorflow та інших Python фреймворків, якщо моделі були створені з допомогою Keras.

Після огляду основних фреймворків для роботи з нейронними мережами на мові Java, було обрано Deeplearning4j через більший набір функцій, підтримуваних архітектур та інтеграцію з Apache Hadoop, Spark і Tensorflow [12].

2.4 Вибір архітектури згорткової нейронної мережі

Для вирішення задачі розробки рекомендаційної системи на основі зображень елементів та з використанням нейронних мереж, можна підібрати та використати вже готову архітектуру, яка найбільш підійде для роботи з графічними зображеннями.

2.4.1 LeNet-5

LeNet-5 є багатошаровою нейронною мережею, і вона навчена алгоритмом зворотного поширення. Ця архітектура була, в основному, спрямована на розпізнавання символів, написаних від руки та на машинці.

Вона має дуже просту архітектурну конструкцію і значно меншу кількість шарів у порівнянні з сучасними глибокими нейронними мережами. Але в поєднанні з правильним оптимізатором і швидкістю навчання це може дати справді хороші результати. Також, за певними даними, мережа є вдалим прикладом техніки навчання на основі градієнта.

LeNet-5 може легко розпізнавати письмові та друковані цифри та документи, навіть якщо шаблони вносять певну мінливість. З часу впровадження LeNet-5, робота, заснована на Computer Vision та Deep Learning, пройшла довгий шлях. Можна легко сказати, що це, мабуть, було визначальним моментом для світу мережевих технік комп'ютерного зору, що породило ще багато проривів [13].

Архітектура складається загалом із 7 шарів, що складаються з 2 наборів шарів згортки та 2 наборів шарів середнього об'єднання, за якими слідує згладжувальний шар згортки. Після цього ми маємо 2 щільні повністю зв'язані шари і, нарешті, класифікатор softmax [14].

Вхідний шар. Якщо взяти стандартне зображення MNIST, тоді отримується вхідне зображення (32x32) у відтінках сірого, яке проходить через перший шар згортки з 6 картами функцій або фільтрами, що мають розмір (5x5) ядра і з кроком 1. Значення вхідних пікселів нормуються так, що білий фон і чорний план переднього плану відповідають -0,1 та 1,175 відповідно, роблячи середнє значення приблизно 0, а дисперсію приблизно 1. Цей вхідний шар не враховується в структурі мережі LeNet-5 зазвичай, та не розглядається як один із ієрархії мережі.

Перший шар (Рис. 2.2). Результат згортки вхідного зображення з 6 фільтрами повинен призвести до зміни розміру від (32x32x1) до (28x28x6), і отримується перший шар. Отже, 1 канал змінюється на 6 каналів, оскільки до вхідного зображення

застосовується 6 фільтрів. Крім того, розмір зображення був зменшений в результаті нульових відступів із розміром ядра (5x5).

Рис. 2.2 Перший шар LeNet-5

У згортковій мережі значення фільтра – це параметри, що отримуються шляхом навчання. Кількість параметрів дорівнює сумі ваг та зміщення, що помножено на кількість фільтрів. Отримуємо такий вираз:

$$num = (w + bi) * f,$$

де num – кількість параметрів тренування, w – ваги, bi – зміщення для кожного фільтра та f – кількість фільтрів. Розрахувавши за формулою, отримаємо $num = (5 * 5 + 1) * 6 = 156$. Загальна кількість з’єднань буде дорівнювати добутку кількості параметрів тренування та площі фільтра - 122304.

Другий шар (Рис. 2.3). У другому шарі реалізовано середній шар об’єднання з розміром фільтра (2x2) і кроком 2. Отже, результуючий розмір зображення зменшиться до (14x14x6). Тут кожна одиниця на кожній карті об'єктів підключена до (2 x 2) сусідів відповідної карти об'єктів у першому шарі.

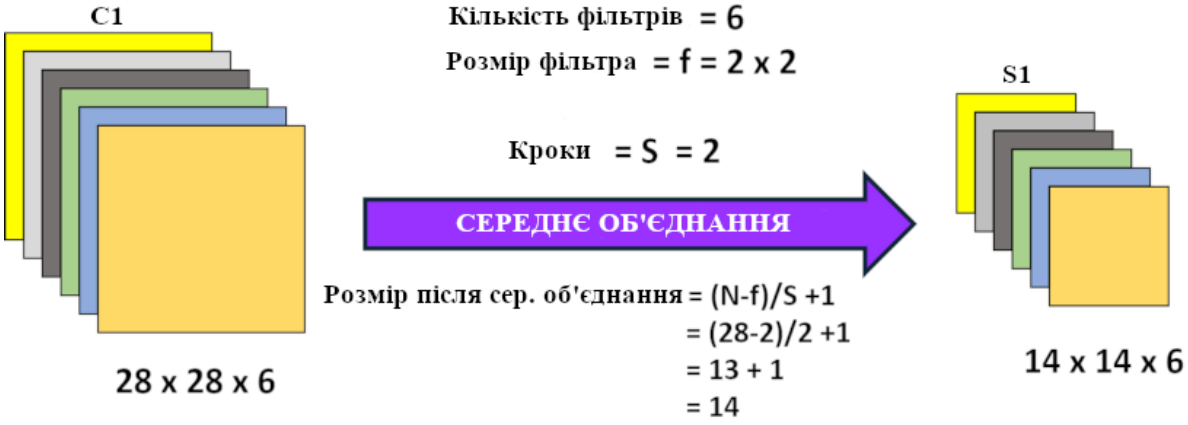

Рис. 2.3 Другий шар LeNet-5

Чотири входи додаються до одиниці в S2 з відповідної карти функцій в C1, потім множаться на тренувальний коефіцієнт і додають до нього зміщується зміщення. Потім результат передається через сигмоїдальну функцію активації, і ми отримуємо результат Q (Рис. 2.4).

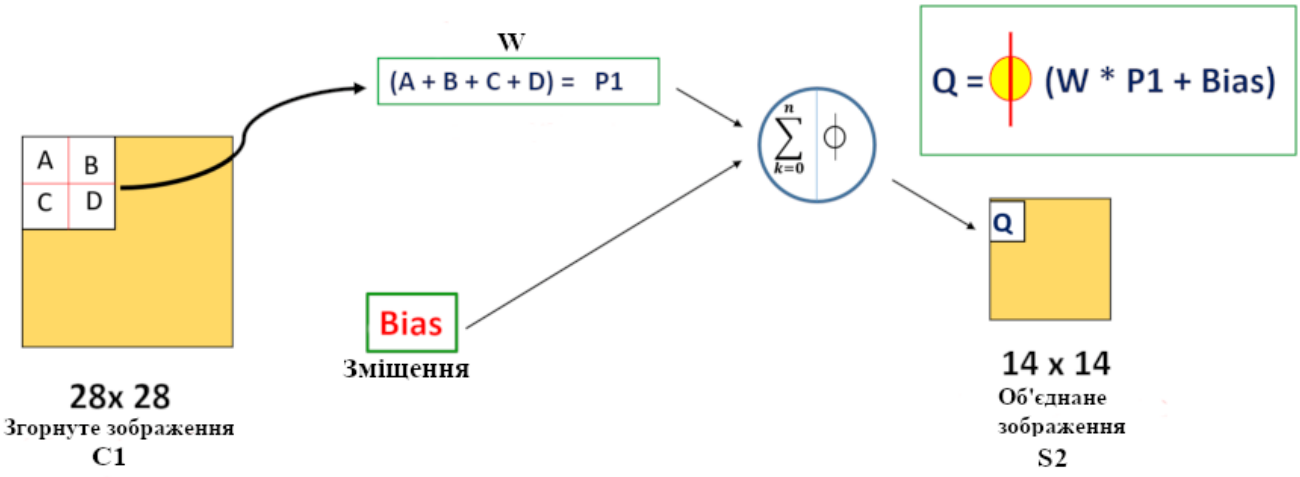

Рис. 2.4 Отриманий результат Q

Третій шар (Рис. 2.5). Якщо перейти до третього шару, застосовується 16 фільтрів з розміром ядра (5x5) до S2, що призводить до рівня згортки C3 з 16 картами функцій. Ця згортка призводить до зміни розміру зображення з (14 x 14 x 6) у S2 на (10 x 10 x 16) у C3.

Рис. 2.5 Третій шар LeNet-5

Як показано на Рис. 2.5, що вхід, тобто S2, має 6 шарів, а вихід, тобто C3, має 16 шарів. Тому ми не можемо безпосередньо зіставити кожен вхідний шар із вихідним шаром. Отже, завдяки цьому кожна одиниця на кожній карті об’єктів, тобто C3, пов’язана з кількома (5 x 5) районами в однакових місцях у підмножині карт об’єктів S2. Поєднання різних вхідних карт функцій із S2 дозволить створювати більше нових функцій.

Четвертий шар (Рис. 2.6). У четвертому шарі знову застосовується середній шар об’єднання з розміром фільтра (2x2) та кроком 2. Отже, результуюче зображення має результуючу середнього пулу, який буде мати розмірність (5x5x16). Тут кожна одиниця на кожній карті об'єктів S4 підключена до (2 x 2) сусідів відповідної карти об'єктів на C3.

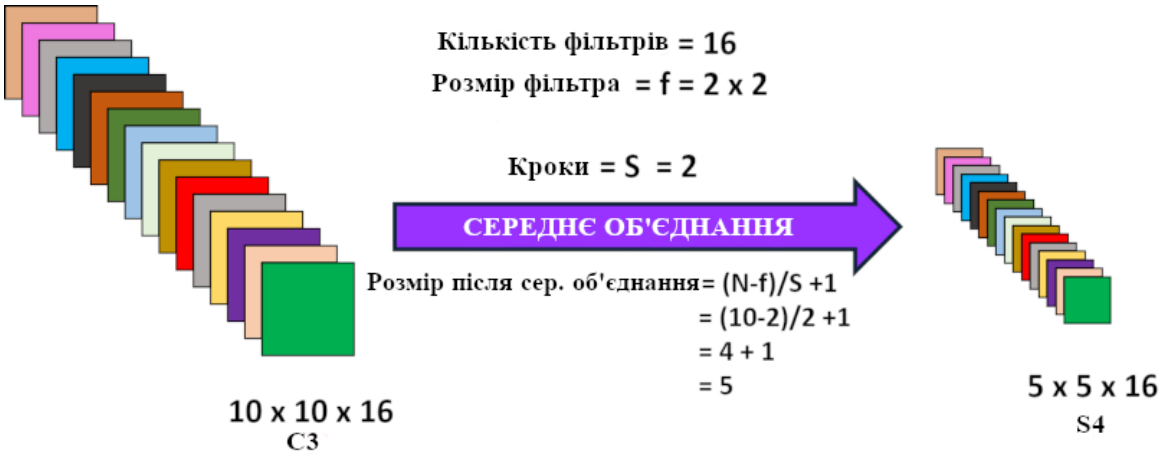

Рис. 2.6 Четвертий шар LeNet-5

П’ятий шар (Рис. 2.7). У п’ятому шарі отримується повністю зв’язаний згортковий шар C5, який має 120 одиниць нейронів, і кожна одиниця C5 підключена до (5 x 5) сусідства на всіх 16 картах функцій S4, тобто кожна одиниця C5 підключена до всіх карти функцій S4 і, отже, C5 відомий як повнозв’язний шар згортки.

Отже, у четвертому шарі отримані розміри становлять (5x5x16), тож загальні вузли становлять 5x5x16 = 400 нейронів. Це означає, що 400 вузлів підключено до 120 вузлів як щільна повністю зв’язна мережа.

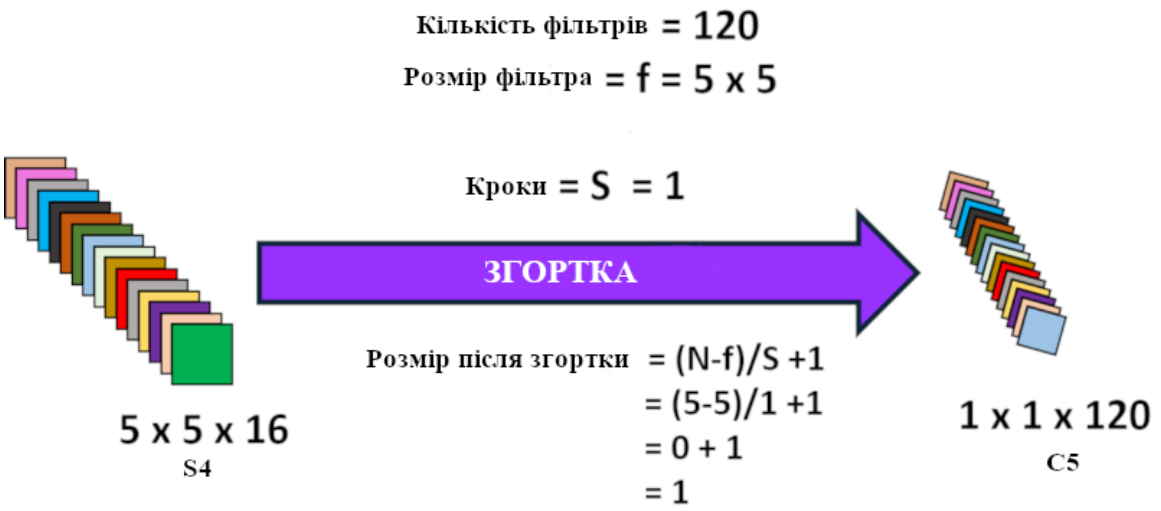

Рис. 2.7 П'ятий шар LeNet-5

Шостий шар (Рис. 2.8). Шостий шар F6 складається з 84 нейронів, повністю пов'язаних з C5. Тут виконується крапковий добуток між вхідним вектором і вектором ваги, а потім до нього додається зміщення. Потім результат передається через сигмоїдальну функцію активації.

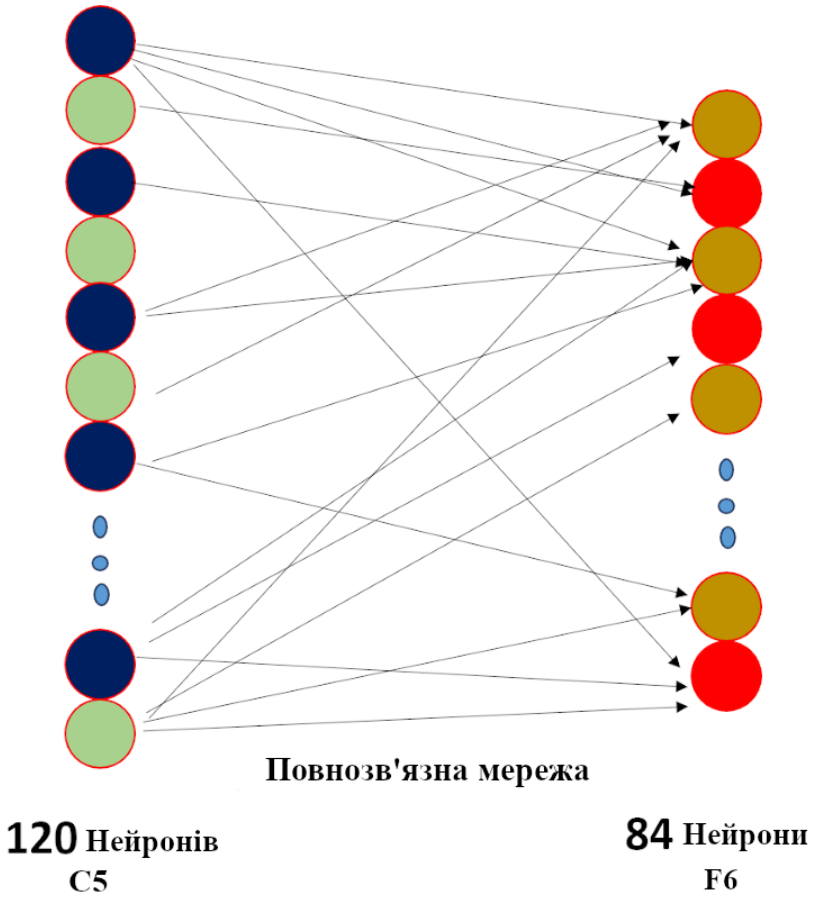

Рис. 2.8 Шостий шар LeNet-5

Кількість нейронів у шарі F6 вибирається як 84, що відповідає растровому зображенню 7 x 12, -1 означає білий, 1 означає чорний, тому чорно-біле бітове зображення кожного символу відповідає коду. Таке подання корисно для розпізнавання рядків символів, взятих із набору для друку ASCII. Символи, які виглядають подібними та заплутаними, як великі літери «O», «0» та малі «o», матимуть однакові вихідні коди.

І нарешті, отримується повністю зв’язний вихідний шар softmax (Рис. 2.9) з 10 можливими значеннями, що відповідають цифрам від 0 до 9.

Отже, отримується функція «активації softmax» на вихідному шарі та інших шарах, які, як було зазначено, мають «$tanh$», оскільки функція активації, оскільки softmax, дасть вірогідність появи кожного вихідного класу в кінці.

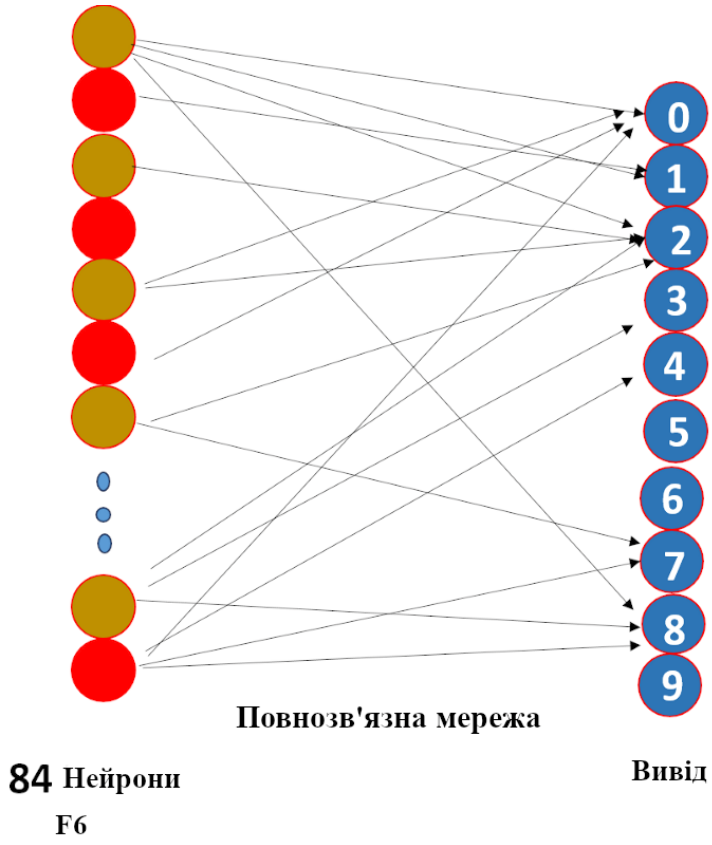

Рис. 2.9 Шар Softmax

2.4.2 AlexNet

AlexNet – це згорткова нейромережа, що мала великий вплив на середовище машинного навчання та алгоритми комп'ютерного зору. Ця мережа виграла конкурс ImageNet LSVRC-2012 у 2012 році з великим відривом 15,3% проти 26,2% (друге місце). Ця мережа має схожі риси з мережею Яна ЛеКуна LeNet. Проте, AlexNet має більшу кількість згорткових шарів та більше фільтрів на кожному шарі. Мережа складається з шарів згортки, максимального пулінгу, дропаут, аугментацію даних та функції активації [15].

Особливості AlexNet**:**

1) Використовується функція ReLu, замість Tanh, щоб додавати нелінійність. Такий підхід прискорює швидкість у 6 разів при збереженні точності.
2) Використовується дропаут замість регуляризації таким чином вирішує проблему перенавчання. Однак, час навчання подвоюється з показником дропауту 0.5.
3) Накладання пулінгу для зменшення розміру мережі. Це зменшує коефіцієнти помилок топ-1 та топ-5 на 0,4% та 0,3% відповідно.

Архітектура мережі складається з восьми шарів з коефіцієнтами. Перші шари згорткові (їх п’ять), а інші три – повнозв’язні. Мережа на виході має функцію активації «softmax» та близько тисячі міток класів. Ядра згорткових шарів 2, 4 та 5 пов’язані безпосередньо з картами ядра попереднього шару, які знаходяться на тому самому графічному процесорі. Ядра третього згорткового шару повністю пов’язані з усіма ядрами попереднього, другого, шару. Повнозв’язні шари мають нейрони, які пов’язані з всіма нейронами попередніх шарів. Дропаути застосовуються перед першим та другим повнозв’язними шарами. Згорткові шари виробляють 95% обчислень, хоч і припадає на них всього 6% параметрів. Архітектура мережі AlexNet зображена на Рис. 2.10.

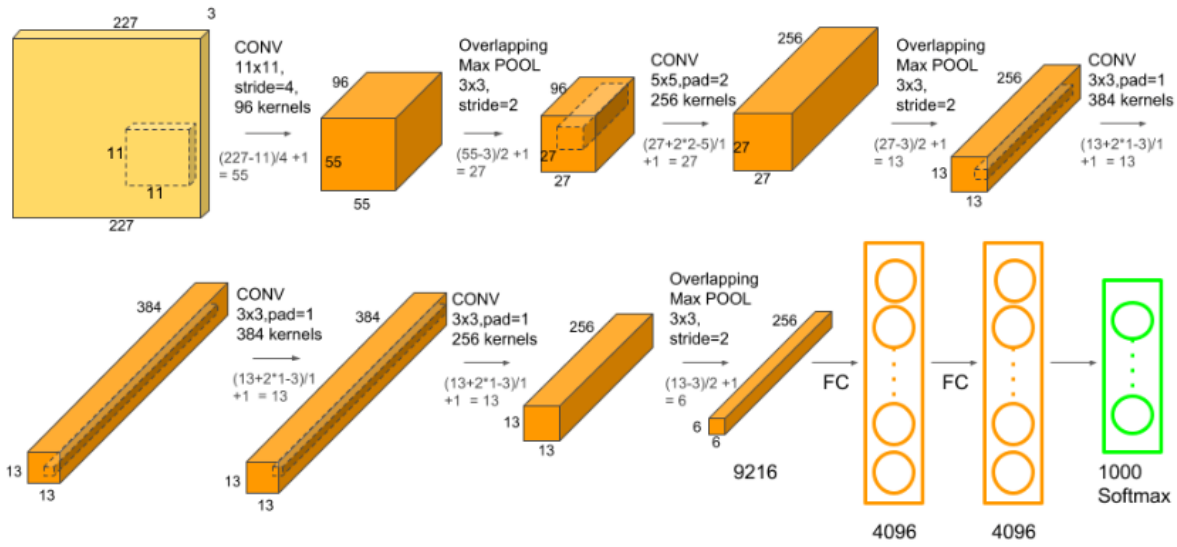

Рис. 2.10 Архітектура мережі AlexNet[15]

2.4.3 VGG-16

VGG-16 - модель згорткової нейромережі, яку запропонували вчені з Оксфордського університету. Модель досягає точності 92.7%. Ця нейромережа являється покращеною версією AlexNet, у якій замінені фільтри(розміру 11 та 5 у першому та другому згорткових шарах відповідно) на декілька фільтрів розміру 3 x 3, які йдуть один за одним (Рис. 2.11).

Вхідними даними в мережу є зображення розмірів (224, 224, 3). Перші два шари мають 64 канали 3 * 3 розміру фільтра і однакові відступи. Потім після максимального шару пулу з кроком (2, 2), два згорткових шари з 256 фільтрами, які мають розмір фільтра (3, 3). Далі слідує максимальний шар об'єднання (2, 2), який збігається з попереднім шаром. Тоді є 2 згорткові шари розміру фільтра (3, 3) і кількістю в 256 фільтрів. Після цього існує 2 набори з 3-х згорткових шарів та максимального шару пулу. Кожен з них має 512 фільтрів розміром (3, 3) з однаковим відступом. Це зображення потім передається в стек двох шарів згортки. У цих шарах згортки та максимального об’єднання використовуються фільтри розміром 3 * 3 замість 11 * 11 у AlexNet та 7 * 7 у ZF-Net. У деяких шарах також використовується 1 * 1 піксель, який використовується для маніпулювання кількістю вхідних каналів. Після кожного згорткового шару виконується заповнення 1 пікселем, щоб запобігти просторовій характеристиці зображення.

Після стеку згорткового шару та шару максимального об’єднання отримується (7, 7, 512) карта функцій. Ми згладжуємо цей вихід, щоб зробити його (1, 25088) вектором ознак. Після цього є 3 повністю зв’язаних шара, перший шар бере вхід з останнього вектора ознак і має на виході вектор (1, 4096), другий шар також має на виході вектор розміру (1, 4096), але третій шар має на виході 1000 каналів, після чого вихід 3-го повністю підключеного шару передається шару softmax для нормалізації вектора класифікації. Після виведення вектора класифікації топ-5 категорій для оцінки. Усі приховані шари використовують ReLU як свою функцію активації. ReLU є більш обчислювально-ефективним, оскільки призводить до швидшого навчання, а також зменшує ймовірність зникнення проблеми градієнта [16].

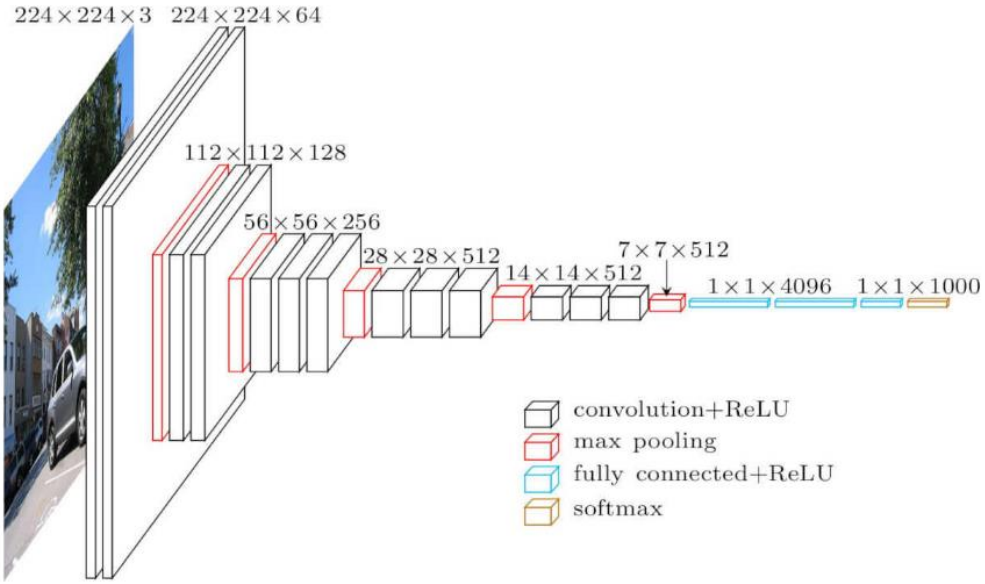

Рис. 2.11 Архітектура нейромережі VGG-16[16]

2.5 Алгоритм визначення рекомендацій

Основний алгоритм визначення рекомендацій було реалізовано, як показано на Рис. 2.12.

Рис. 2.12 Алгоритм визначення рекомендацій

Для того, щоб проаналізувати усі зображення системи, потрібно запустити окремий аналізатор, який до початку роботи інтернет-ресурсу зможе проаналізувати усі зображення, що знаходяться у системі.

Аналізатор повинен завантажити усі зображення та обробити їх перед початком аналізу. Їх потрібно привести до стандартної форми та перевести у нормалізований векторний вигляд. Після цього зображення повинні пройти через нейромережу. Під час проходження заздалегідь навчена нейромережа аналізує зображення та виділяє вихідну карту характеристик. Вона має вигляд великого вектору з числами. Для того, щоб під час кожного запуску аналізатора не виділяти карту характеристик, можна скористатися базою даних або будь-яким хмарним сховищем.

Після аналізу окремих зображень потрібно розрахувати подібність між усіма зображеннями попарно. Тобто, кожне зображення аналізується у парі з іншим і вираховується косинусоїдальна відстань між векторами їх характеристик. Вихідний коефіцієнт показує наскільки вектори подібні, а отже і самі зображення подібні між собою.

На основі отриманих коефіцієнтів потрібно створити матрицю (таблицю), у якій стовпці та рядки будуть визначати зображення, а дані у матриці – коефіцієнти їх схожості. Таку таблицю можна зберігати як локально у вигляді CSV файлу, так і у NoSQL базі даних, або у хмарному сховище.

Головний алгоритм завершено. Після роботи аналізатора можна запускати сервіс, який використовує проаналізовані зображення. Коли користувач, наприклад, інтернет-магазину обирає товар і переходить на сторінку з деталями, то система аналізує «матрицю схожості». Відбувається пошук за назвою зображення, або унікальним ідентифікатором рядка чи стовбця, що відповідає цьому зображенню і надалі рядок/стовпець сортується за спаданням. Таким чином, коефіцієнти будуть розставлені від найбільш схожого до менш схожого. Далі серед відсортованих значень виділяється певна кількість перших значень і відправляється до клієнтської частини додатку. Додаток робить пошук товарів за ідентифікаторами і представляє їх у частині рекомендованих товарів.

2.6 Аналіз ефективності методу рекомендацій на основі методу аналізу зображень нейромережею

Для того, щоб рекомендувати товари на основі візуальної складової, дуже часто постає проблема точності обробки зображень та часу виконання операцій. Такі проблеми виникають, здебільшого, через час, який потрібно витратити на нормалізацію зображення та приведення його до вигляду, який буде зручно обробити.

Точність обчислення залежить від алгоритму виявлення характеристик оцінюваного зображення та порівняння цих характеристик з існуючими для інших зображень. Для того, щоб точніше виявити схожість, потрібно використовувати багато обчислень, щоб перевірити всі можливі перетворення зображення.

Рекомендаційна система на основі аналізу графічних зображень з допомогою нейромережі дозволяє значно скоротити час через збереження вектору характеристик, з якими працює нейромережа, до бази даних, або іншого сховища. Також точність здобувається тим, що архітектура згорткової нейромережі направлена на визначення особливостей кожного зображення та збереження їх до вектора характеристик, що допомагає при знаходженні косинусоїдальной відстані між векторами.

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділі були розглянуті актуальні мови програмування та середовища розробки задля вибору найбільш зручної для вирішення поставленої задачі. Ними стали мова Java для серверної частини, Angular для клієнтської та середовище розробки Intellij Idea.

Було детально розглянуто різні архітектури згорткових нейромереж, оцінені їх переваги та недоліки, а також фреймворки, з допомогою яких буде вестися робота з нейромережею. Фреймворком був обраний Deeplearning4j через можливість програмування на мові Java та архітектурою була обрана VGG-16, так як показує найкращий результат в обробці та аналізі зображень. Був розроблений алгоритм рекомендацій на основі методу аналізу зображень нейромережею та збереженням попередніх характеристик до бази даних, а також була проаналізована його ефективність.

3 РОЗДІЛ
РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧІ ТА РОЗРОБКА ПРОГРАМИ

3.1 Вимоги до програмного продукту

Для демонстрації роботи запропонованого алгоритму, було вирішено розробити веб додаток у вигляді інтернет-магазину та додатковий модуль, що окремо від додатку аналізує зображення та розраховує рекомендації. Розроблюваний додаток повинен бути прости для розуміння роботи алгоритму, та забезпечувати виконання наступних вимог:

1. Можливість завантаження вхідних зображень для аналізу.
2. Відображення роботи рекомендаційної системи, тобто самих рекомендацій.

Для виконання поставленого завдання було обрано мову Java з використанням SpringBoot Framework для розробки веб додатку. Також, для розробки клієнтської частини було обрано Angular Framework [17], що забезпечує візуальний інтерфейс. Для реалізації роботи з нейромережею буде використано фреймворк Deeplearning4j, що надасть можливість використовувати обрану раніше архітектуру VGG16 та датасет ImageNet [18], щоб аналізувати зображення та виділяти карти їх характеристик. Для того, щоб будувати матрицю рекомендацій зображень, буде використано фреймворк Tablesaw [19].

Для коректної роботи програми, було визначено наступні оптимальні характеристики ПК:

- Чотирьох-ядерний процесор з тактовою частотою 1.8ГГц
- 16ГБ RAM
- Операційна система Windows 7 або новіше
- Встановлена Java версії 8 та новіше

3.2 Реалізація методу визначення рекомендацій товарів на основі графічних зображень

Отже, метод має такі етапи:

1) Завантаження даних для роботи нейромережі: завантаження архітектури VGG16 нейромережі та датасету ImageNet

2) Робота з зображеннями: завантаження зображень для інтернет магазину, нормалізація зображення, завантаження зображення до нейромережі та його обробка, отримання карти характеристик зображень.

3) Побудова матриці схожості: побудова матриці на основі векторів схожості зображень.

4) Використання рекомендаційної системи: збереження файлу з рекомендаціями та робота додатку на основі цих рекомендацій.

3.2.1 Реалізація кроку завантаження даних для роботи нейромережі

Для роботи з нейромережею було обрано фреймворк Deeplearning4j. Перед безпосередньою роботою з нейромережею проходить настроювання середовища та завантаження архітектури мережі, завантаження датасету. Так, як було обрано архітектуру VGG16, то на початковому етапі потрібно створити екземпляр класу VGG16 з допомогою конструктора new VGG16().

Після створення екземпляра класу VGG16(), потрібно завантажити датасет. Фреймворк Deeplearning4j має вбудовані можливості для завантаження датасетів, зокрема ImageNet. Після того, як усе було завантажено та додаток готовий для наступних кроків, можна переходити безпосередньо до роботи з графічними зображеннями та їх аналізу.

3.2.2 Реалізація роботи з зображеннями

На початку потрібно завантажити усі зображення товарів, що існують у певному інтернет-магазині. В даному випадку, вони будуть зберігатися у локальному сховищі комп’ютера, але можна розширити можливості та користуватися хмарними сховищами.

Безпосередня робота із зображеннями виконується у окремому модулі додатку, що називається «*analyzer».* Для запуску всіх етапів існує метод *analyzeImage()* у контролері *AnalyzeController.java* (Рис. 3.1).

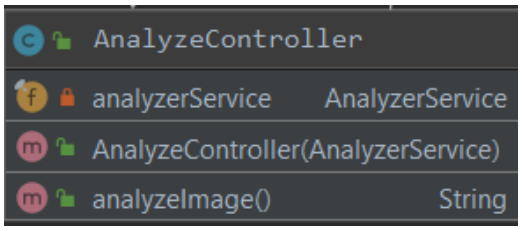

Рис. 3.1 Структура класу AnalyzeController.java

Після відправки запиту на роботу аналізатора, робота переходить до сервісу, що називається *AnalyzerService.java* (Рис. 3.2).

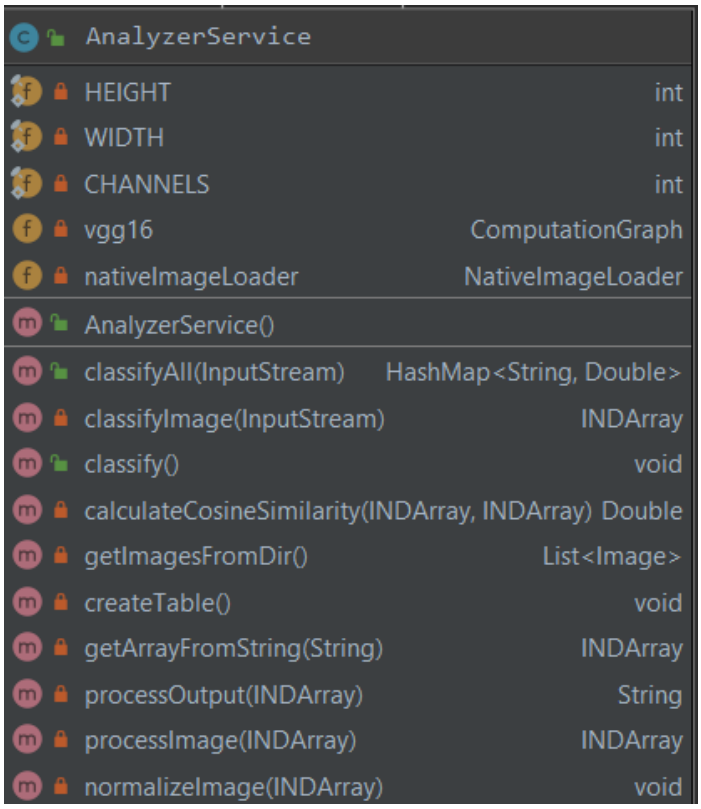

Рис. 3.2 Структура класу AnalyzerService.java

У методі *classify()* відбувається головна робота з аналізу зображень. Перш за все потрібно завантажити зображення з відповідної директорії. Це відбувається у методі *getImagesFromDir()*. Під час завантаження зображення одразу і відбувається його приведення до певних розмірів, нормалізація та виділення карти характеристик, що відбувається у методі *classifyImage()*.

Після відпрацювання методу *getImagesFromDir()* отримується список об’єктів класу *Image.java* (Рис. 3.3). Цей список передається далі для формування матриці.

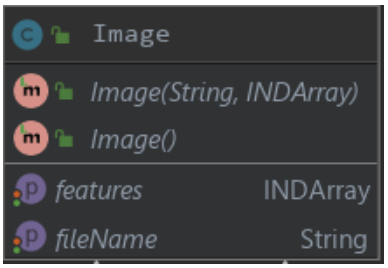

Рис. 3.3 Структура класу Image.java

3.2.3 Реалізація побудові матриці схожості зображень

Після попереднього етапу отримується список усіх зображень з їх іменами, векторами характеристик. Далі потрібно визначити схожість кожного зображення з усіма іншими та на основі таких даних збудувати матрицю схожості. Для обчислення схожості використовується розрахунок косинусоїдальної відстані. Ці обчислення відбуваються безпосередньо під час запису до матриці, що відбувається у методі *classify()*.

Матриця будується з допомогою розширення Tablesaw, методом побудови таблиці. У таблиці кожен стовпець має назву, що означає назву зображення, кожен рядок також має назву, що відповідає назві зображення. На перетині двох зображень записується коефіцієнт]х схожості. Чим більший коефіцієнт, тим більш схожі зображення. Така таблиця зберігається до сховища у вигляді CSV (Comma-separated values) і у подальшому може використовуватись у інтернет-магазину для відображення рекомендованих товарів. Вигляд такої таблиці представлений на Рис. 3.4.

Names	100627.255.jpg	100627.72.jpg	100657.216.jpg	100657.72.jpg	101026.3.jpg	101093.316189.jpg
100627.255.jpg	1.0000001192092896	0.6509705185890198	0.7230314016342163	0.9813668727874756	0.19591616094112396	0.3384196162223816
100627.72.jpg	0.6509705185890198	1.0000001192092896	0.9943576455116272	0.7837669253349304	0.09814756363630295	0.050667136907577515
100657.216.jpg	0.7230314016342163	0.9943576455116272	1	0.8406450152397156	0.13209053874015808	0.0654468908905983
100657.72.jpg	0.9813668727874756	0.7837669253349304	0.8406450152397156	1.0000001192092896	0.17378439009189606	0.3036505877971649
101026.3.jpg	0.19591616094112396	0.09814756363630295	0.13209053874015808	0.17378439009189606	0.9999998807907104	0.049298934638500214
101093.316189.jpg	0.3384196162223816	0.050667136907577515	0.0654468908905983	0.3036505877971649	0.049298934638500214	1
101093.316195.jpg	0.28409329056739807	0.0433029904961586	0.0508989579975605	0.2577313184738159	0.03952940180897713	0.9974818825721741
101093.316313.jpg	0.30951374769210815	0.04962284490466118	0.0602668896317482	0.27994081377983093	0.04690499231219292	0.9991153478622437
101093.342648.jpg	0.4136618673801422	0.06346085667610168	0.08817244321107864	0.3677385449409485	0.06763225048780441	0.9949641227722168
101175.325.jpg	0.11816132068634033	0.06421640515327454	0.09479667246341705	0.1000659167766571	0.8676121830940247	0.026965316385030746
...	...	...	...	...	...	...
New_Balance_FuelCore_NERGIZE.jpg	0.3832981586456299	0.17795373499393463	0.20170390605926514	0.3545081317424774	0.07415226846933365	0.1620936244726181
northside_bishop.jpg	0.3893651068210602	0.04410843178629875	0.09125307202339172	0.33062413334846497	0.34881383180618286	0.3419061005115509
puma_rs_2.0_base.jpg	0.05474783480167389	0.028747785836458206	0.022897476330399513	0.04745616018772125	0.00492260605096817	0.007989364676177502
PUMA_Vikky_V2_grey.jpg	0.03568084165453911	0.007808435708284378	0.0014488425804302096	0.026543304324150085	0.0014039831003174186	0.0009879356948658824
PUMA_Vikky_V2_red.jpg	0.03804329037666321	0.00831577554345131	0.002214117906987667	0.02857164666056633	0.0020560205448418856	0.0022197654470801353
PUMA_Vikky_V2_violet.jpg	0.03537658974528313	0.007734321989119053	0.0013604722917079926	0.026268301531672478	0.001356207299977541	0.0002365042018936947
stride_rite_theo.jpg	0.6270231604576111	0.15831516683101654	0.21474255621433258	0.5514991283416748	0.13257725536823273	0.42592257261276245
stride_rite_theo_silver.jpg	0.0438886359333992	0.01675834320485592	0.010602631606161594	0.03550038859248161	0.0034894137643277645	0.0050798095762729645
western_bonnie.jpg	0.7881680130958557	0.06635051965713501	0.15892133116722107	0.6643621921539307	0.1546097844839096	0.5571929812431335
western_chief_rain.jpg	0.03031664527952671	0.012121976353228092	0.035978663712739944	0.015279927290976048	0.8223341703414917	0.0003158278123009 95

Рис. 3.4 Таблиця схожості зображень

3.2.4 Реалізація використання рекомендацій

Основний додаток інтернет-магазин має клієнтську частину та серверну. Під час відкриття сторінки деталей певного товару відбувається запит з клієнтської частини додатку до серверної, у якому надсилається назва поточного зображення. Серверна частина додатку має сервіс *RecomendationService.java*, що проводить аналіз значень, що збережені у csv файлі та обирає товари, що можуть рекомендуватися. Його структура наведена на Рис. 3.5.

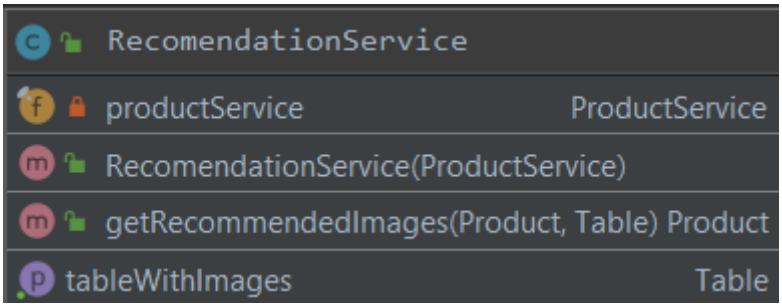

Рис. 3.5 Структура класу RecomendationService.java

Саме метод *getRecommendedImages()* аналізує коефіцієнти та обирає зображення, що найбільш схожі на поточне. У додатку існує тип Product.java (Рис. 3.6), що відображає структуру продукту і має у собі поле *recommended* для збереження

рекомендованих продуктів. Після аналізу рекомендацій, у це поле записуються рекомендовані продукти та цей об’єкт повертається до клієнтської частини, де відображається.

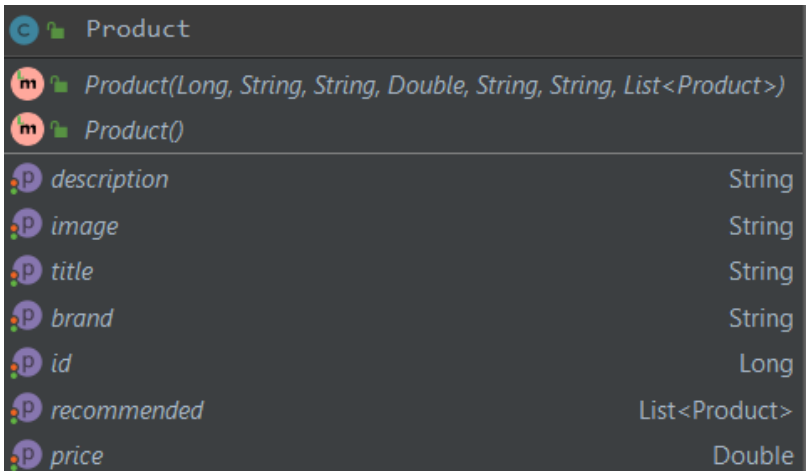

Рис. 3.6 Структура класу Product.java

3.3 Інструкція користувачеві

Розроблюваний додаток має вигляд інтернет-магазину, що працює на локальному комп’ютері на порту 5000 – серверна частина, а клієнтська частина доступна на порту 4200. Щоб відкрити сторінку, потрібно відкрити будь-який браузер та ввести у адресному рядку localhost:4200.

На даному етапі додаток має декілька екранів. Головний екран виглядає як каталог інтернет-магазину, у якому знаходяться товари. Це видно на Рис. 3.7.

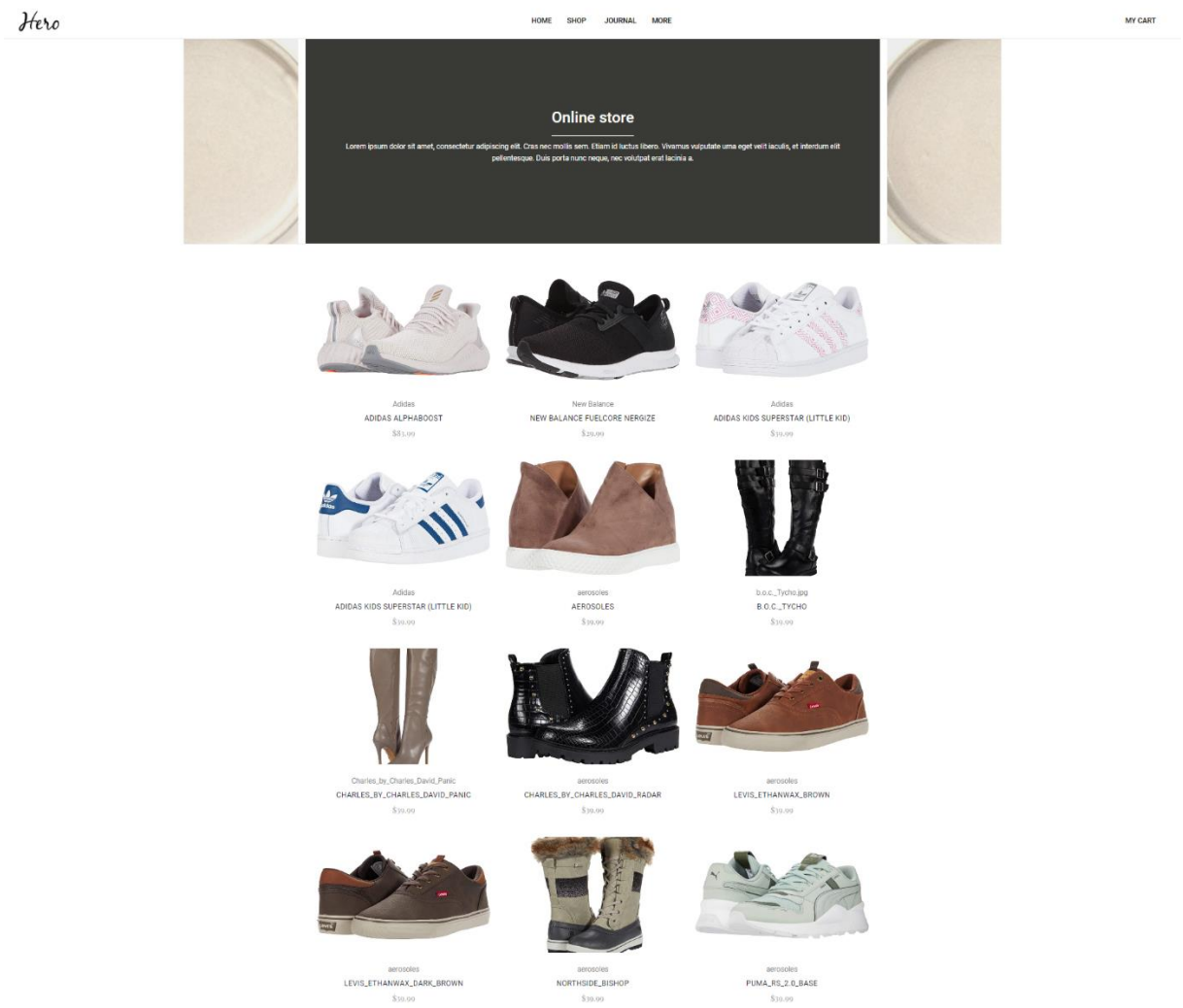

Рис. 3.7 Головна сторінка додатку

Для того, щоб побачити рекомендовані товари, потрібно обрати будь-який товар з каталогу та навести мишкою на обраний товар. З’явиться меню, у якому будуть опції «Показати деталі» та «Додати у кошик». Для відкриття товару потрібно натиснути «Показати деталі» (Рис. 3.8.).

Рис. 3.8 Опції при наведенні на товар

Після відкриття деталей товару буде відкрита безпосередньо сторінка з товаром та будуть показані рекомендовані товари у нижній частині сторінки (Рис. 3.9).

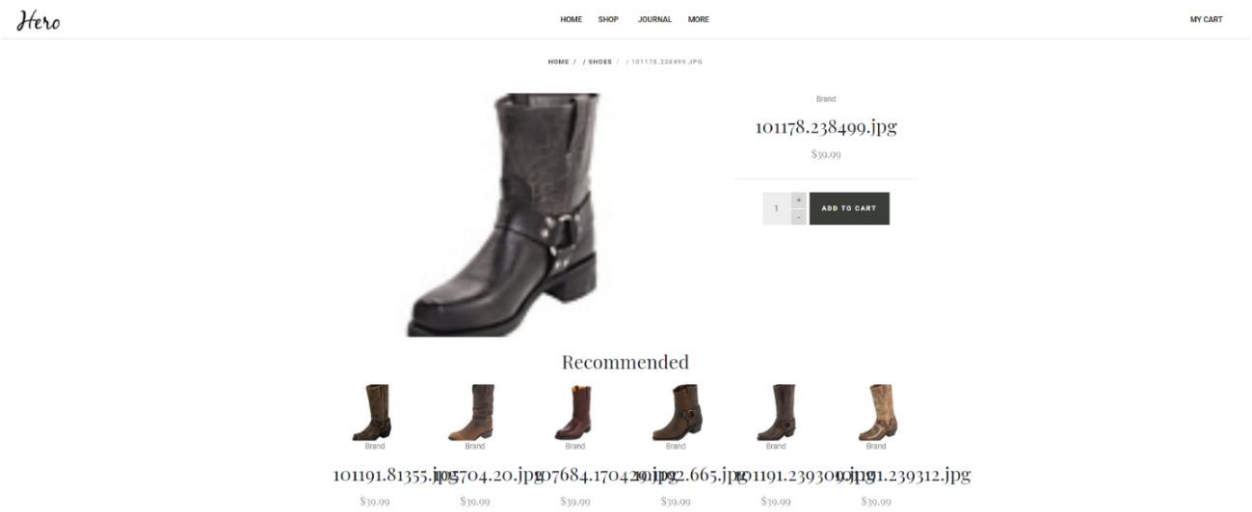

Рис. 3.9 Сторінка з деталями товару

У поточній версії додатку усі категорії взуття беруть участь у аналізі і не приймаються до уваги під час прогнозування, тому іноді можуть бути серед рекомендованих товарів не дуже коректні (Рис. 3.10). Для більш точного рекомендування потрібно розширити додаток категоризуванням рекомендацій.

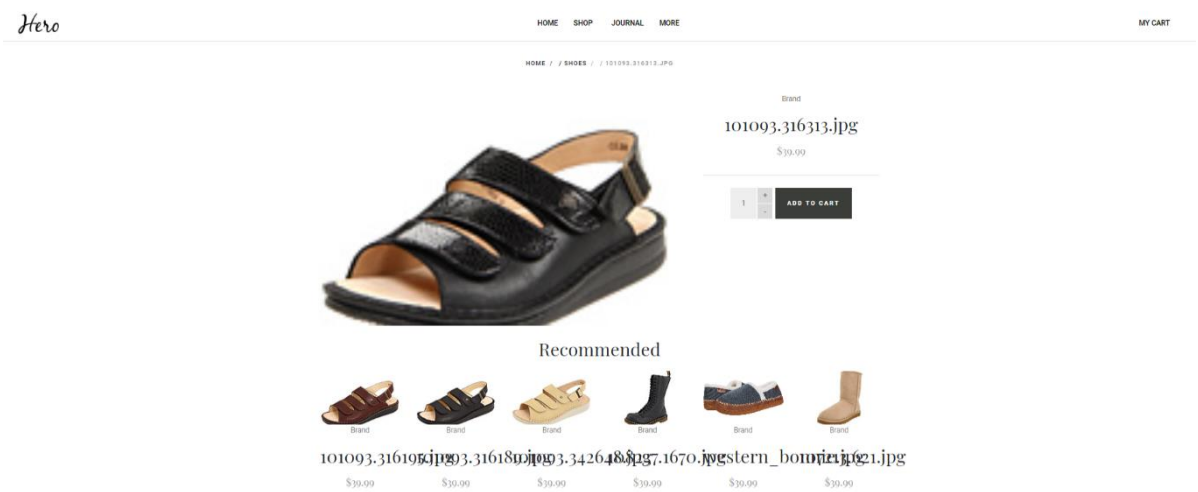

Рис. 3.10 Сторінка з деталями товару

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було розглянуто реалізацію алгоритму пошуку рекомендованих товарів на основі зображень з допомогою технології нейронних мереж. Були розглянуті усі етапи роботи алгоритму та детально описані. Також були описано деталі інтеграції з нейронною мережею, робота з фреймворком Deeplearning4j та Tablesaw. Після опису алгоритму була надана детальна інструкція користувача, що демонструє працездатність заданого алгоритму. Даний алгоритм перш за все вирішує проблему «холодного старту», при якій не надано жодних персоналізованих даних про користувачів та їх уподобання. Додаток за заданим алгоритмом пошуку рекомендацій справляється з поставленою задачею та для більш точних результатів може бути розширений та покращений.

4 РОЗДІЛ
РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Рекомендаційні системи з кожним роком все більш застосовуються у різних сервісах. Їх використання допомагає бізнесу збільшувати дохід та більш комфортно для користувача надавати будь-які послуги, або продавати товари. Від якості рекомендаційної системи та її типу залежить наскільки прибутковим буде сервіс, або інтернет-магазин.

Найпопулярніші сервіси використовують технології рекомендаційних систем, що аналізують усі дані про користувача. Це можуть бути і просто оцінки користувачем товарів, його коментарі, товари, які він уже купував та супутні з ними, так і навіть просто карта переходів по сайту. Такі дані потрібно збирати тривалий час та аналізувати і порівнювати з іншими користувачами сервісу, або магазину. Коли ж сервіс тільки запускається, то даних для аналізу немає і постає проблема «холодного старту». Ідея даного проекту дозволяє вирішити цю проблему доволі ефективно та оптимально.

В даному дипломному проекті реалізовано тестову версію такої рекомендаційної системи та додатку у вигляді інтернет-магазину, що використовує розроблену рекомендаційну систему для прикладу працездатності даного підходу. В цьому розділі буде описано стартап-проект підходу до рішення проблеми «холодного старту» у рекомендаційних системах з допомогою аналізу зображень товарів з використанням технології нейронних мереж. В Табл. 4.1 викладено зміст ідеї, що пропонується, основні переваги, які може отримати користувач з ідеї та чим відрізняється даний підхід від існуючих підходів.

Таблиця 4.1

Опис ідеї, напрямків застосування та вигод користувача

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Рекомендаційна система, яка аналізує графічні зображення, з використанням технології нейронних мереж	Вирішення проблеми «холодного старту»	Можливість отримувати рекомендації товарів з самого початку запуску сервісу
	Точна рекомендація на основі візуального відображення	Дозволяє бачити рекомендовані товари, що найбільш схожі за візуальною частиною на поточний товар
	Визначення дуплікації зображення товару	Можливість попередити повторне завантаження того самого зображення товару

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників).

Попереднім колом аналогів підходів є групові рекомендації (group recommendation to individual user), фільтрботи (filterbots), експертний шлях.

Таблиця 4.2

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Мій проект	Аналоги		
			1	2	3
1	Можливість застосовувати автоматичний підхід	+	+	+	-
2	Наявність певної кількості базових користувачів	-	+	+	-
3	Високий рівень точності схожості рекомендації з поточним товаром	+	-	-	-
4	Відсутність потреби в особистих даних користувача	+	-	-	-
5	Рекомендація супутніх товарів	-	+	+	+
6	Аналіз візуальної складової товарів	+	-	-	-

Визначений перелік слабких, сильних та нетральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. Як видно з Табл. 4.2, основними перевагами є відсутність необхід-

ності мати базові статистичні дані користувачів та відсутність потреби у зборі особистих даних. Основним недоліком є те, що немає можливості рекомендувати супутні товари, але даний підхід перед усім вирішує проблему «холодного старту».

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Рекомендаційна система, яка аналізує графічні зображення, з використанням технології нейронних мереж	Мова програмування Java	Є в наявності	Доступні безкоштовно
2		Фреймворк Deeplearning4j	Є в наявності	Доступні безкоштовно
3		Архітектура нейронної мережі та датасет ImageNet	Є в наявності	Доступні безкоштовно
4		Фреймворк Tablesaw	Є в наявності	Доступні безкоштовно

Обрана технологія реалізації ідеї проекту: розробка програми на мові програмування Java, фреймворк для роботи з нейронними мережами, фреймворк для роботи з таблицями та CSV файлами, архітектура нейронної мережі та датасет ImageNet, в тому числі запропонований в даному дипломному проекті.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє визначити та окреслити напрями розвитку проекту із урахуванням стану потокового ринку, потреб потенційних клієнтів та пропозицій проектів-

конкурентів. В Табл. 4.4 наведена попередня характеристика потенційного ринку для розроблюваного стартап-проекту.

Таблиця 4.4

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	>10
2	Динаміка ринку (якісна оцінка)	В сфері рекомендаційних систем ринок знаходиться на етапі стрімкого зросту
3	Наявність обмежень для входу (вказати характер обмежень)	Велика кількість конкурентних підходів та наявність у великих гравців уже настроєної бази користувачів, що відкидає проблему «холодного старту»
4	Специфічні вимоги до стандартизації та сертифікації	Немає вимог

Таблиця 4.5

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Рекомендація товарів з відсутністю початкових користувачів	Нові сервіси та інтернет-магазини, що тільки запускають свою роботу та не мають достатньої кількості користувачів для аналізу та прогнозування рекомендацій	Кількість товарів та відповідно зображень, які будуть піддаватися аналізу	Необхідність коректних та доволі точних рекомендацій товарів прямо від початку запуску інтернет-магазину
2	Відсутність збирання персональних даних	Інтернет-магазини та сервіси з продажу товарів.	Кількість товарів та відповідно зображень, які будуть піддаватися аналізу	Відсутність необхідності збирати персональні дані користувача та наявність рекомендацій у разі використання «анонімного акаунту», тобто доступу до сервісу без входу до профілю

Таблиця 4.6

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкуренція	На даний момент існує багато рішень, що можуть вирішить проблему «холодного старту», деякі з них плавно перетворюються в основну рекомендаційну систему	Швидка реакція команди розробників на побажання потенційних клієнтів, можливість розширення даного рішення та його доповнення.
2	Вузький функціонал та спектр можливостей	На ринок потрібно виходити з рішенням, яке максимально буде покривати усі можливі проблеми галузі для того, щоб бути конкурентноспроможним	Можлива інтеграція до якогось сервісу з метою колаборації та скорішого виходу на ринок, також можливе кардинальне розширення функціоналу
3	Брак коштів	Через те, що проект новий та фінансове становище обмежене, можливий брак коштів, адже компанія ще не дає достатнього прибутку	Залучення інвесторів з добре розробленим бізнес-планом або пропозиція колаборації з сервісами з продажу

Таблиця 4.7

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Залучення інвестицій	Для залучення інвестицій потрібно розробити чіткий якісний бізнес план та можливості розширення і розвитку	Погодитися на співпрацю з іншими компаніями, залучити спеціалістів, які допоможуть у створенні стратегії розвитку
2	Розширення клієнтської бази	За рахунок створення чіткого бізнес-плану та стратегії розвитку, також можливої співпраці з провідними сервісами можна розширити базу потенціальних клієнтів	Запропонувати колаборацію відомим сервісам, або тим, ідея яких була популярна ще до виходу на ринок. Також можна проводити презентації продукту
3	Заключення контрактів на співпрацю	Заключення контрактів з потенційними клієнтами для розгортання розробленого сервісу та розширення існуючого функціоналу у рамках індивідуального підходу до кожного клієнта	Запропонувати співпрацю на, можливо, безоплатній основі задля більшого рекламування продукту

Таблиця 4.8

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Складність входу на ринок	Багато рекомендаційних систем інтегруються у сервіси з допомогою розробників самого сервісу і зачасту не потребують залучення сторонніх компаній та спеціалістів	Пропонувати більш вигідні умови розробки. Це означає як і фінансову сторону, так і можливість рішення «під ключ», що прискорить інтеграцію. Також можливе розширення функціоналу, що привабить потенційного клієнта
Загальнодоступність підходів	Алгоритми підбору рекомендацій знаходяться у загальному доступі, тому багато систем можуть їх інтегрувати	Важливо пропонувати не тільки загальнодоступні алгоритми, а і рішення їх оптимізації та можливість швидкого та комфортного інтегрування з поточною системою клієнта

Таблиця 4.9

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкурентні підходи в галузі	Потенційні конкурентні підходи в галузі	Постачальники	Клієнти	Товари-замінники
	Групові рекомендації (group recommendation to individual user), фільтр-боти (filterbots), експертний шлях та інші	Рішення рекомендацій на основі властивостей товарів	Популярні сервіси та інтернет-магазини	Сервіси з продажу товарів та інтернет-магазини	Open Source рішення, залучення експертів та гібридні рішення
Висновки:	Існує велика кількість можливих рішень проблеми «холодного старту»	Існує безліч підходів, що можуть мати схожі властивості оцінки	Постачання таких систем частіше відбувається напряму всередині компанії між командами розробників	Більшість сервісів уже інтегрували у себе системи рекомендацій, що вдосконалюються дуже швидко, але дане рішення розраховано на «нових гравців» ринку	Зачасту проблема «холодного старту» вирішується формуванням рейтингу товарів з допомогою експерта або ж генерування рейтингу з допомогою гібридних рішень

Таблиця 4.10

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспромож-ності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Простий інтерфейс для роботи з додатком	Так як продукт ще на стадії розробки та проектування, то існує можливість проаналізувати існуючі варіанти інтерфейсів для роботи з додатком, або розробити такий, що буде задовольняти більшість користувачів на ринку
2	Забезпечення зручної інтеграції	Часто рекомендаційні системи інтегруються напряму у існуючий додаток та доволі важко змінювати поточний алгоритм рекомендації. Даний підхід передбачає створення окремого модулю, що дозволить більш комфортно користуватися такою системою
3	Аналіз «фідбеку» від користувачів	Під час розробки продукту важливою складовою є аналіз потреб користувача, тому можна розширити функціонал додатку для можливості забезпечення відгуків, пропозицій та питань. Такий підхід дозволить максимально швидко розширяти функціонал та робити продукт актуальним

Таблиця 4.11

Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспромо-жності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Простий інтерфейс для роботи з додатком	11					✓		
2	Забезпечення зручної інтеграції	18						✓	
3	Аналіз «фідбеку» від користувачів	10				✓			

Таблиця 4.12

SWOT- аналіз стартап-проекту

Сильні сторони: Зручний інтерфейс для роботи з додатком Забезпечення зручної інтеграції Можливість аналізу конкурентних підходів Аналіз побажань та оцінок клієнтів	**Слабкі сторони:** Велика кількість сервісів з уже інтегрованими рішеннями Наявність opensource рішень Недовіра великих клієнтів Невелика кількість «нових гравців» на ринку, для яких буде цікава ідея
Можливості: Отримати пропозицію для співпраці та розвитку продукту	**Загрози:** Рішення клієнтів самостійно розробляти своє рішення

Таблиця 4.13

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розміщення ідеї на відкритому ресурсі	Низька ймовірність через відсутність достатнього фінансування та можливість закриття проекту на етапі тестової версії	Упродовж тижня
2	Розповсюдження проекту через конференції, публічні виступи	Середня ймовірність, але вона є, так як через конференції потенційні клієнти можуть дізнатися та про наявний продукт та запропонувати варіанти співпраці	Від півроку і більше
3	Укладання договорів з новими сервісами на ринку	Більша за середню, адже самостійно вийти на ринок доволі важко, а у співпраці з більш стабільним сервісом або інтернет-магазином зробити це буде простіше. Співпраця можлива на безоплатній основі з перспективою розділення прибутку	Близько півроку, або у залежності від рівня готовності інших компаній до співпраці

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.14

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Сервіси з продажу товарів	Не всі користувачі готові	Продукт не є необхідністю, але покращує роботу сервісів і може приваблювати користувачів	Висока конкуренція	Доволі складний вхід
2	Інтернет-магазини	Готові прийняти усі	Продукт необхідний усім інтернет-магазинам через можливість збільшення потенційних клієнтів магазину	Висока конкуренція	Вхід помірно складний
3	Інші стартап-проекти	Готові прийняти продукт	Можуть бути зацікавлені у використанні даного продукту	Конкуренція середня	Вхід не є складним

Які цільові групи обрано:

Можна працювати з усіма цільовими групами потенційних клієнтів, але слід надавати перевагу приватним клієнтами, таким як сервіси з продажу товарів та інтернет-магазини. Слід обирати такі групи потенційних споживачів через більш вірогідний прибуток від співпраці та більший шанс вигоди від реклами за рахунок співпраці.

Таблиця 4.15

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розвиток шляхом інтернет-маркетингу та демонстрації продукту на публічних заходах	Вплив на ключових цільових споживачів та рекламування продукту	Презентація використаних підходів до створення продукту з точки зору науки та запровадження рішення у інших стартап-проектах	Оперативно розширювати та оновлювати продукт відповідно до побажань та вимог клієнтів

Таблиця 4.16

Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні, проект не є «першопроходьцем»	Можна рухатися паралельно, шукаючи нових клієнтів та намагатися «перемани-ти» споживачів у конкурентів	Через схожість ідеї буде відбуватися копіювання головних характеристик з можливим особистим доповненням	Пропонувати оптимальніше розв’язання проблеми та забезпечити комфортне використання розробленого продукту

Таблиця 4.17

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто- спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Зручність використання, простота та «легкість» рішення, функціонал, що задовольняє потреби	Залучити інших спеціалістів для генерування нових ідей розширення і доповнення продукту	Оперативне реагування на потреби користувача і відповідне оновлення функціоналу	Рекомендаційна система, рекомендації на основі графічних зображень, рішення проблеми «холодного старту»

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Рекомендація товарів у інтернет-магазині	Реалізація рекомендаційної системи для інтернет-магазинів, що продають товари на основі аналізу їх графічних зображень	Можливість аналізувати графічні зображення, що мають товари та достатньо точно рекомендувати схожі за виглядом товари
2	Подолання проблеми «холодного старту»	Вирішення проблеми «холодного страту», що виникає на старті роботи сервісів через брак статистичних даних для рекомендації товарів користувачам	Розрахунок рекомендацій товарів з мінімальним використанням даних та розрахункових потужностей. Для аналізу потрібні лише зображення
3	Відсутність необхідності використовувати персональні дані користувачів	Можливість рекомендувати товари у інтернет-магазині та не примушувати користувача залишати свої особисті дані для розрахунку рекомендацій	До уваги не беруться ніякі персональні дані користувачів, що дозволяє працювати системі навіть у режимі «анонімного профілю», тобто без необхідності заходити до особистого профіля користувача

Таблиця 4.19

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Рекомендаційна система, що базується на аналізі графічних зображень товарів з інтернет-магазину та використовує технології нейронних мереж.</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Модуль, що реалізує аналізатор зображень сервісу та продукує рекомендації</td><td>1.15МБ (без урахування графічних зображень, що обробляються)</td></tr>
<tr><td>2. Модуль для тесту системи, а саме приклад реального інтернет-магазину, що використовує розроблену рекомендаційну систему</td><td>410МБ (з урахуванням зображень, що репрезентують товари у інтернет-магазині)</td></tr>
<tr><td colspan="2">Якість: внутрішнє тестування додатку, система обробки помилок та логування непередбачених ситуацій</td></tr>
<tr><td colspan="2">Пакування: продукт доступний при завантаженні з офіційного сайту, або напряму налаштовує спеціаліст</td></tr>
<tr><td colspan="2">Марка: назва організації-розробника та назва товару</td></tr>
</table>

Продовження таблиці 4.19

III. Товар із підкріпленням	До продажу: програмний код
	Після продажу: ліцензія з можливістю підтримки спеціалістами та можливостями розширення
За рахунок чого потенційний товар буде захищено від копіювання: пряме налаштування програмного продукту спеціалістами компанії без доступу до програмного коду, ключі до ліцензії	

Таблиця 4.20

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Інтеграція сервісу рекомендаційної системи напряму до системи цільового клієнта з допомогою спеціалістів компанії	Має надавати кваліфікованих спеціалістів, що будуть проводити інтегрування продукту з поточною системою клієнта	Передбачено одноріневий канал збуту через прямий контракт з потенційним споживачем	Право власності залишається у розробника продукту

Таблиця 4.21

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	Дані не розповсюджуються	Дані не розповсюджуються	>10000$/місяць	150-300

Таблиця 4.22

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Заключать контракт на співпрацю напряму під час розробки, або перед запуском сервісу	Зустрічі на конференціях, реклама за рахунок безоплатних колаборацій	Продаж програмного забезпечення та консультування з питань рекомендаційних систем	Демонстрація ключових характеристик продукту та переваг перед альтернативними рішеннями	Охоплення потенційних споживачів та презентація головних характеристик продукту, що виділяє його серед альтернативних рішень

ВИСНОВКИ ДО РОЗДІЛУ 4

Даний розділ містить інформацію про розробку стартап-проекту до розроблюваного програмного продукту. Також було наведено опис ідеї проекту, проведено аналіз конкурентів ринку, їх можливостей, було розглянуто сильні та слабкі сторони і показані особливості, що виділяють розроблюваний проект на фоні інших аналогів.

У відповідних підрозділах було описано наступні особливості проекту:

- Було наведено головну ідею проекту, описано функціонал розроблюваного програмного забезпечення, наведено основні конкурентні рішення та підходи і порівняно їх характеристики для визначення особливостей, що виділяють розроблюваний проект серед інших.
- Проведено технологічний аудит розроблюваного програмного забезпечення та визначення особливостей його реалізації.
- Порівняно конкурентні рішення у розрізі ринкових можливостей, перспективи запуску програмного продукту. Також розроблена стратегія виходу на ринок, визначено цільову групу споживачів та проаналізовано їх вимоги до продукту.
- Було проаналізовано канали збуту, описані основні принципи просування та способи продажу продукту, визначено шляхи та кроки компанії, які можуть бути вжиті для охоплення більшої аудиторії та скорішого виходу на ринок.

Отже, після аналізу було отримано стартап-проект, що відображає шлях виходу програмного продукту на ринок, його запуск. Також отримано навички у створенні стартап-проектів, аналізу ринку конкурентних рішень, визначення маркетингової стратегії виходу на ринок та аналізу особливостей проекту, що виокремить продукт на фоні інших.

ВИСНОВКИ

Дана магістерська робота була розроблена з метою вирішення задачі побудови оптимізованого рішення до створення рекомендаційної системи, яка аналізує графічні зображення, що використовуються для наповнення контентом сторінок товарів сервісів з продажу або інтернет-магазинів. Для вирішення поставленої задачі було використано технології нейронних мереж, що показують високі результати під час аналізу графічних зображень.

У відповідних розділах роботи було розглянуто базові теоретичні відомості про технологію нейронних мереж, а саме про згорткові нейромережі, було розглянуто види їх архітектури та проаналізовано шари кожного типу нейронної мережі. Було наведено відомості про рекомендаційні системи, особливо про проблему «холодного старту» та розглянуто шляхи її подолання.

Під час розробки програмного рішення було розглянуто середовища розробки програмного забезпечення, мови програмування та допоміжні технології для створення web-додатків, їх підтримки та запуску. Було обрано фреймворк, що задовольняє усім потребам та має найбільш повний функціонал для роботи з нейронними мережами, їх архітектурами та даними, що будуть використані при аналізі графічних зображень. Для збереження проміжних даних, а саме рекомендацій, також було обрано бібліотеку, що має задовільний набір функцій. Після огляду різних архітектур згорткових нейронних мереж, було обрано для роботи VGG-16 архітектуру через найкращі показники тесту при роботі з графічними зображеннями.

Розроблено алгоритм, за яким працює рекомендаційна система, що аналізує графічні зображення та розглянуто ефективність і особливості цього алгоритму. Наведено реалізацію даного алгоритму і описані відповідні розроблені модулі системи, класи та методи. Було детально описано механізм роботи з нейронною мережею, підготовки даних та кожен етап аналізу зображень для побудови рекомендацій.

У відповідному розділі було запропоновано інструкцію користувача, що показує головні кроки для відображення працездатності програмного забезпечення. Розроблений програмний продукт реалізує наведений алгоритм побудови рекомендацій, що будуються на основі аналізу графічних зображень та використанням технології нейронних мереж.

ЛІТЕРАТУРА

1. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms [Електронний ресурс]. Режим доступу: http://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf (дата звернення: 30.06.2020)
2. Memory-based algorithms [Електронний ресурс]. Режим доступу: http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/memorybased.html (дата звернення: 30.06.2020)
3. Model-based recommendation systems [Електронний ресурс]. Режим доступу: http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html (дата звернення: 01.07.2020)
4. David J. Livingstone. Artificial Neural Networks. Methods and Applications [Електронний ресурс]. Режим доступу: https://dl.uswr.ac.ir/bitstream/Hannan/130691/1/MMB vol.458 Livingstone D.J. (eds.) Artificial Neural Networks. Methods and Applications (Humana Press, 2010)(ISBN 1588297187)(259s).pdf (дата звернення: 10.07.2020)
5. Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi [Електронний ресурс]. Режим доступу: https://link.springer.com/content/pdf/10.1007/s13244-018-0639-9.pdf (дата звернення: 20.08.2020)
6. IntelliJ IDEA features [Електронний ресурс]. Режим доступу: https://www.jetbrains.com/idea/features/ (дата звернення: 15.09.2020)
7. Eclipse IDE [Електронний ресурс]. Режим доступу: https://www.eclipse.org/ (дата звернення: 15.09.2020)
8. Netbeans Features [Електронний ресурс]. Режим доступу: https://netbeans.apache.org/ (дата звернення: 15.09.2020)
9. Building REST services with Spring [Електронний ресурс]. Режим доступу: https://spring.io/guides/tutorials/rest/ (дата звернення: 16.09.2020)

10. Сравнение стеков Java EE и Spring: возможности и ограничения [Електронний ресурс]. Режим доступу: https://dou.ua/lenta/articles/javaee-vs-spring/ (дата звернення: 16.09.2020)
11. Neuroph. About Neuroph project [Електронний ресурс]. Режим доступу: http://neuroph.sourceforge.net/about_project.html (дата звернення: 01.10.2020)
12. Deeplearning4j. Guide [Електронний ресурс]. Режим доступу: https://deeplearning4j.org/docs/latest/ (дата звернення: 01.10.2020)
13. LeNet-5: A Practical Approach [Електронний ресурс]. Режим доступу: https://debuggercafe.com/lenet-5-a-practical-approach/ (дата звернення: 15.10.2020)
14. The Architecture and Implementation of LeNet-5 [Електронний ресурс]. Режим доступу: https://medium.com/towards-artificial-intelligence/the-architecture-implementation-of-lenet-5-eef03a68d1f7 (дата звернення: 15.10.2020)
15. A Walk-through of AlexNet [Електронний ресурс]. Режим доступу: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637 (дата звернення: 20.10.2020)
16. VGG-16 | CNN model [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/vgg-16-cnn-model/ (дата звернення: 22.10.2020)
17. Angular Framework Features [Електронний ресурс]. Режим доступу: https://angular.io/features (дата звернення: 20.10.2020)
18. ImageNet [Електронний ресурс]. Режим доступу: http://www.image-net.org/ (дата звернення: 15.10.2020)
19. Tablesaw framework [Електронний ресурс]. Режим доступу: https://jtablesaw.github.io/tablesaw/gettingstarted.html (дата звернення: 30.10.2020)
20. Рекомендательная система: введение в проблему холодного старта [Електронний ресурс]. Режим доступу: https://habr.com/ru/company/surfingbird/blog/168733/ (дата звернення: 15.11.2020)

ДОДАТКИ

ДОДАТОК А

Оптимізована рекомендаційна система з використанням технології нейронних мереж

Текст програми

Листів 12

Київ – 2020

```
package com.project.analyzer.controller;

import com.project.analyzer.service.AnalyzerService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import java.io.IOException;

@Controller
public class AnalyzeController {
    private final AnalyzerService analyzerService;
    public AnalyzeController(AnalyzerService analyzerService) {
        this.analyzerService = analyzerService;
    }
    @GetMapping("/classify")
    public String analyzeImage() throws IOException {
        analyzerService.classify();
        return "Classified!";
    }
}
```

```
package com.project.analyzer.service;

import com.project.analyzer.model.Image;
import org.datavec.image.loader.NativeImageLoader;
import org.deeplearning4j.nn.graph.ComputationGraph;
import org.deeplearning4j.zoo.PretrainedType;
import org.deeplearning4j.zoo.ZooModel;
import org.deeplearning4j.zoo.model.VGG16;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.dataset.api.preprocessor.DataNormalization;
import org.nd4j.linalg.dataset.api.preprocessor.VGG16ImagePreProcessor;
import org.nd4j.linalg.factory.Nd4j;
import org.nd4j.linalg.ops.transforms.Transforms;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import tech.tablesaw.api.DoubleColumn;
import tech.tablesaw.api.StringColumn;
import tech.tablesaw.api.Table;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Stream;

@Service
public class AnalyzerService {
    private static final int HEIGHT = 224;
    private static final int WIDTH = 224;
    private static final int CHANNELS = 3;
    private ComputationGraph vgg16;
    private NativeImageLoader nativeImageLoader;

    @Autowired
    public AnalyzerService() {
        try {
            //Setup the VGG16 model from the DL4J ModelZoo
            ZooModel zooModel = VGG16.builder().build();
            vgg16 = (ComputationGraph) zooModel.initPretrained(PretrainedType.IMAGENET);
        } catch (IOException e) {
            e.printStackTrace();
        }
        nativeImageLoader = new NativeImageLoader(HEIGHT, WIDTH, CHANNELS);
    }
    public HashMap<String, Double> classifyAll(InputStream file) {
        INDArray image = null;
        try {
            image = nativeImageLoader.asMatrix(file);
        } catch (IOException e) {
            e.printStackTrace();
        }
        normalizeImage(image);
```

```
        INDArray output = processImage(image);
        HashMap<String, Double> res = new HashMap<>();
        return res;
    }
    private INDArray classifyImage(InputStream file) {
        INDArray image = null;
        try {
            image = nativeImageLoader.asMatrix(file);
        } catch (IOException e) {
            e.printStackTrace();
        }
        normalizeImage(image);
        return processImage(image);
    }
    public void classify() throws IOException {
        List<Image> processedImages = getImagesFromDir();
        List<String> imageNames = new ArrayList<>();
        processedImages.forEach(image -> imageNames.add(image.getFileName()));
        Table analizedImages =
                Table.create("analizedImages")
                        .addColumns(
                                StringColumn.create("Names", imageNames.toArray(new String[0])));
        for (int i = 0; i < processedImages.size(); i++) {
            List<Double> features = new ArrayList<>();
            for (Image processedImage : processedImages) {
                features.add(calculateCosineSimilarity(processedImages.get(i).getFeatures(),
processedImage.getFeatures()));
            }
            analizedImages.insertColumn(i + 1,
DoubleColumn.create(processedImages.get(i).getFileName(), features.toArray(new Double[0])));
        }
        analizedImages.write().csv("C:/Users/alytvy/Documents/recomendation-sys/recomendation-
sys/analyzed-images.csv");
        System.out.println(analizedImages);
    }
    private Double calculateCosineSimilarity(INDArray columnImage, INDArray rowImage) {
        return Transforms.cosineSim(columnImage, rowImage);
    }
    private List<Image> getImagesFromDir() {
        List<Image> files = new ArrayList<>();
        try (Stream<Path> paths = Files.walk(Paths.get("C:/Users/alytvy/Documents/recomendation-sys/" +
                "recomendation-sys/imgs/shoes"))) {
```

```
            paths
                    .filter(Files::isRegularFile)
                    .forEach(file -> {
                        try {
                            files.add(new Image(file.getFileName().toString(), classifyImage(new
FileInputStream(file.toFile()))));
                        } catch (FileNotFoundException e) {
                            e.printStackTrace();
                        }
                    });
        } catch (IOException e) {
            e.printStackTrace();
        }
        return files;
    }
    private INDArray getArrayFromString(String features) {
        String[] splitted = features.split(" ");
        double[] arr = new double[splitted.length];
        for (int i = 0; i < splitted.length; i++) {
            arr[i] = Double.parseDouble(splitted[i]);
        }
        INDArray res = Nd4j.create(arr);
        return res;
    }
    private String processOutput(INDArray array) {
        String output = array.toString();
        output = output.substring(1, output.length() - 1);
        output = output.replaceAll(",", " ");
        output = output.replaceAll(" +", " ");
        return output;
    }
    private INDArray processImage(final INDArray image) {
        INDArray[] output = vgg16.output(true, image);
        return output[0];
    }
    /**
     * Normalize the image
     *
     * @param image
     */
    private void normalizeImage(final INDArray image) {
```

```
        DataNormalization scaler = new VGG16ImagePreProcessor();
        scaler.transform(image);
    }
}
```

```
package com.project.analyzer.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.nd4j.linalg.api.ndarray.INDArray;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Image {
    private String fileName;
    private INDArray features;
}
```

```
package com.project.analyzer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AnalyzerApplication {

    public static void main(String[] args) {
        SpringApplication.run(AnalyzerApplication.class, args);
    }
}
```

```
package com.recomendation.demo.controller;

import com.recomendation.demo.entity.Product;
import com.recomendation.demo.service.ProductService;
import com.recomendation.demo.service.RecomendationService;
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import tech.tablesaw.api.Table;

import java.io.IOException;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    private final ProductService productService;
    private final RecomendationService recomendationService;

    public ProductController(ProductService productService, RecomendationService recomendationService)
{
        this.productService = productService;
        this.recomendationService = recomendationService;
    }

    @GetMapping(value = {"", "/"})
    public Iterable<Product> getProducts() {
        return productService.getAllProducts();
    }

    @GetMapping(value = "/{id}")
    public Product findByIds(@PathVariable Long id) throws IOException {
        Table analyzedImages = recomendationService.getTableWithImages();
        Product product = productService.getProduct(id);
        return recomendationService.getRecommendedImages(product, analyzedImages);
    }
}
package com.recomendation.demo.service;

import com.recomendation.demo.entity.Product;
import org.springframework.stereotype.Service;
import tech.tablesaw.api.Table;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

@Service
```

```
public class RecomendationService {
    private final ProductService productService;

    public RecomendationService(ProductService productService) {
        this.productService = productService;
    }
    public Table getTableWithImages() throws IOException {
        return Table.read().csv("analyzed-images.csv");
    }
    public Product getRecommendedImages(Product product, Table table) {
        String imageName = product.getImage();
        Table descending = table.sortDescendingOn(imageName).first(7);
        descending.retainColumns("Names", imageName);
        List<Product> products = new ArrayList<>();
        for (int i = 0; i < 6; i++) {
            Double feature = descending.row(i + 1).getDouble(1);
            String pictureUrl = descending.row(i + 1).getString(0);
            products.add(productService.getProductByImage(pictureUrl));
        }
        product.setRecommended(products);
        System.out.println(descending);
        return product;
    }
}
```

```
import {Component, OnInit} from '@angular/core';
import {ActivatedRoute} from "@angular/router";
import {ProductService} from "../../services/products.service";
import {Product} from "../../model/product";
import {CartService} from "../../services/cart.service";

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.css']
})
export class ProductComponent implements OnInit {
  private sub;
  public product: Product;
  public recommendedProd: Product[];
  quantity: number = 1;
```

```
  constructor(private route: ActivatedRoute,
              private productService: ProductService,
              private cartService: CartService
  ) {
  }

  ngOnInit() {
    this.route.params
      .subscribe(res => {
        this.recommendedProd = [];
        this.getProduct(res.id);
      })
  }

  getProduct = (id) => {
    this.sub = this.productService.getProductById(id).subscribe(product => {
        this.product = JSON.parse(product['_body']);
      },
      (error) => console.log(error)
    );
  };

  changeQuantity = (newQuantity: number) => {
    this.quantity = newQuantity;
  };
  addToCart = (product) => {
    if (this.quantity) this.cartService.addToCart({product, quantity: this.quantity})
  };

  ngOnDestroy() {
    this.sub.unsubscribe();
  }
}

<div *ngIf="product" class="product-page">
  <div class="product-page-container">
    <ol class="product-breadcrumbs">
      <li class="breadcrumb-item"><a routerLink="/">Home</a></li>
      <li class="breadcrumb-item"> / <a routerLink="/">Shoes</a></li>
      <li class="breadcrumb-item"> / {{product.title}}</li>
```

```
    </ol>

    <div class="row">

      <div class="col-md-8">

        <div class="product-details-image" [ngStyle]="{'background-image': 'url(./assets/shoes/' +
product.image + ')'}"></div>

      </div>

      <div class="col-md-4">

        <div class="product-details-row">

          <div class="product-brand">{{product.brand}}</div>

          <h1 class="product-title">{{product.title}}</h1>

          <div class="product-price">{{product.price | currency :'USD':true }}</div>

          <div class="product-description">{{product.description}}</div>

        </div>

        <div class="product-details-button">

          <quantity-control [quantity]="quantity" (onChange)="changeQuantity($event)"></quantity-
control>

          <div class="product-cart-button button button-primary button-large"
(click)="addToCart(product)">Add to cart</div>

        </div>

      </div>

    </div>

    <div class="product-title">Recommended</div>

    <div class="row">

      <div class="col-md-2" *ngFor="let recom of product.recommended;let i = index">

        <div class="product-image" [ngStyle]="{'background-image': 'url(./assets/shoes/' + recom.image
+ ')'}"></div>

        <div class="product-brand">{{recom.brand}}</div>

        <h1 class="product-title" [routerLink]="['../../product',recom.id]">{{recom.title}}</h1>

        <div class="product-price">{{recom.price | currency :'USD':true }}</div>

      </div>

    </div>

  </div>

</div>


import { Component, OnInit } from '@angular/core';

import {ProductService} from "../../services/products.service";

import {Product} from "../../model/product";

import {CartService} from "../../services/cart.service";

import {Router} from "@angular/router";

import {Subscription} from "rxjs/internal/Subscription";


@Component({

  selector: 'app-category',
```

```
  templateUrl: './category.component.html',
  styleUrls: ['./category.component.css']
})
export class CategoryComponent implements OnInit {
  public products:Array<Product>;
  private sub;
  constructor(
    private productService:ProductService,
    private cartService:CartService,
    private router: Router,
  ) { }

  ngOnInit() {
    this.load();
  }
  load = () => {
    this.sub = this.productService.getAllProducts()
      .subscribe(products => {
          this.products = JSON.parse(products['_body']);
        },
        (error) => console.log(error)
      );
  };
  addToCart = (product) => {
    this.cartService.addToCart({product,quantity:1})
  };
  ngOnDestroy() {
    this.sub.unsubscribe();
  }
}


<div class="header-image">
  <div class="header-block">
    <div class="header-text">
      <div class="header-text-title">Online store</div>
      <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras nec mollis sem. Etiam id luctus
libero.
        Vivamus vulputate urna eget velit iaculis, et interdum elit pellentesque. Duis porta nunc
neque, nec volutpat erat lacinia a.
      </p>
    </div>
```

```
  </div>
</div>
<div class="layout-container container">
  <div class="product-grid row">
    <div class="col-md-4 col-sm-6 col-lg-4 col-xl-4" *ngFor="let product of products;let i = index">
      <div class="image-container">
        <div class="product-image" [ngStyle]="{'background-image': 'url(./assets/shoes/' +
product.image + ')'}"></div>
        <div class="overlay">
          <div class="button button-primary" [routerLink]="['../product',product.id]">View
Details</div>
          <div class="button button-primary" (click)="addToCart(product)">Add To Cart</div>
        </div>
      </div>
      <div class="product-details">
        <div class="product-brand">{{product.brand}}</div>
        <div class="product-title">{{product.title}}</div>
        <div class="product-price">{{product.price | currency :'USD':true }}</div>
      </div>
    </div>
  </div>
</div>
```